Aalto University
School of Engineering

Elizaveta Shishova

Development of an automated simulation environment for Body in White joining techniques

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering

November 2017

Supervisor: Prof. Jani Romanoff

Thesis advisors: Prof. Roger A. Sauer, Dr. Tobias Luginsland, M. Sc. Mohamed Malek

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of master's thesis

Author Elizaveta Shishova

Title of thesis Development of an automated simulation environment for Body in White joining techniques

Degree programme Mechanical Engineering

Major/minor Mechanics of materials **Code** IA3027

Thesis supervisor Prof. Jani Romanoff

Thesis advisor(s) Prof. Roger A. Sauer, Dr. Tobias Luginsland, M. Sc. Mohamed Malek

Date 24.11.2017 **Number of pages** 57 + 7 **Language** English

Abstract

Parts in the joining station are usually arranged and clamped to result in an assembly that perfectly resembles the CAD0 construction. The perfect set-up is usually determined with the help of numerical optimization. In reality there are imperfections specific for each part, which are usually adjusted in the workshop manually. To do numerical optimization for each part is computationally expensive. Here the need for a more efficient Data mining analysis method arises. The present thesis investigates the metamodeling techniques in order to approximate the geometry response to the change of the joining station parameters. The aim is to provide the evaluation of the model by interpolating between the geometry and station set-up variations.

The suitable regressions, which are able to deal with non-linear geometry behaviour, are selected with the help of the literature research. The theory behind the numerical simulation, regressions and sampling is studied to ensure the right choice of the metamodel. An automated simulation environment is programmed to assist with the creation of variation in geometry and joining station, numerical solution and analysis. The chosen regressions - Support Vector Regression and Kernel Ridge Regression - are tested on models of different complexity. The errors are evaluated to provide the quantitative measure of the quality of regressions. Possible improvements of the metamodels are studied, such as Latin Hypercube sampling techniques.

The reduced complexity metamodels proved to provide a good approximation, while dealing well with non-linearities. On the other hand, it was shown that models with many design variables require improvement in sampling technique to provide better result within reasonable computational costs. At the same time, Latin Hypercube did not provide visible advancements in the tested cases. The Automated Simulation Environment and the tested metamodels are a base for the future implementation of an Artificial Neural Networks for defining the perfect set-up of Body in White joining stations for imperfect parts.

Keywords geometrical variation, metamodeling, nonlinear regression, Support Vector Regression, Kernel Ridge Regression

Preface and Acknowledgements

The study was conducted for Daimler AG during April - September 2017 in Sindelfingen, Germany, as a part of Artificial Intelligence training for the Body in White joining simulation techniques project.

I would like to express the appreciation to all the people, who were involved in my Master's thesis project: from Aalto University, RWTH Aachen University and Daimler AG side. I would like to thank my advisors Tobias Luginsland and Mohamed Malek for trusting me with such a challenging topic, all the learning experiences and great support. I am really grateful to the whole Production Process simulation team of TecFabrik of Daimler AG for the very friendly working environment. I would also like to thank Prof. Roger A. Sauer for leading me into the project and all the knowledge of Computational Contact Mechanics. I would like to acknowledge a special gratitude to my supervisor Prof. Jani Romanoff for all the advices and the revision of my work, as well as deep understanding of my topic despite distant communication. All the love to my family and friends who supported me during all of my studies.

Stuttgart, October 2017

Elizaveta Shishova

Contents

Symbols and abbreviations **vi**

1 Introduction **1**

1.1 Problem formulation 1

1.2 Aim of the present work 2

1.3 Limitations 3

1.4 State-of-the-art 3

2 Theoretical framework **6**

2.1 Governing equations 6

2.2 FEM formulation 8

2.3 Contact formulation 11

2.4 Data Mining - Metamodelling techniques 13

2.4.1 Kernel trick 14

2.4.2 Support Vector Regression 15

2.4.3 Kernel Ridge Regression 16

2.4.4 Sampling 17

3 Software framework for automated simulations **18**

3.1 Tools 18

3.1.1 Python 18

3.1.2 LS-DYNA 19

3.1.3 DFS tool 19

3.2 Joining station setup and computational aspects 20

3.3 Input-Output handling 22

3.3.1 Geometry and clamping variation 24

3.3.2 Create cases structure 25
3.3.3 Solve cases structure 27
3.3.4 Analyze cases structure 27

4 Results **29**
4.1 Testing set up 29
4.2 One design variable metamodel 30
4.3 Two design variables metamodel 33
4.4 Two design variables metamodel using LHC 36
4.5 18 design variables metamodel 38

5 Discussion **39**

6 Summary **41**

7 Future work **43**

Appendices **50**

A Python libraries **51**

B Input for the automated environment **52**

C Files for Data Base **54**

D Comparison of regression parameters **55**

Symbols and abbreviations

Symbols

$\mathbf{a}$	acceleration vector
$\mathbf{a_1}, \mathbf{a_2}$	tangential basis vectors
$\mathbf{b}$	body force
B	body domain
$\mathbf{B}$	strain displacement matrix
C	regularization constant for SVR
E	Young's modulus
$\mathbf{f}$	force vector
$\mathbf{g_C}$	contact gap
$\mathbf{g_N}$	contact normal gap
$\mathbf{g_T}$	contact tangential gap
$\mathbf{I}$	identity matrix
k	penalty coefficient (spring constant)
$\mathbf{K}$	stiffness matrix
$\mathbf{M}$	mass matrix
$\mathbf{n}$	normal vector
$\mathbf{N}$	shape function
$\mathbf{P}$	Piola-Kirchhoff stress tensor

R^2	coefficient of determination
t	time
$\mathbf{t}$	traction force vector
$\mathbf{u}$	displacement field
$\mathbf{V}$	volume
$\mathbf{x}$	input set of variables in regression (vector for one variable, matrix for more)
$\mathbf{y}$	response vector in regression
δ_{ij}	Kronecker delta
ϵ	epsilon tube for SVR
$\boldsymbol{\varepsilon}$	strain tensor
κ	kernel function
λ	shrinkage parameter for Ridge regression
μ	second Lamé constant, shear modulus
ν	Poisson ratio
ρ	density
ρ_0	density in initial configuration
$\boldsymbol{\sigma}$	stress tensor
σ^2	bandwidth for RBF
$\boldsymbol{\omega}$	test/weighting function
Π	work
Λ	first Lamé constant

Abbreviations

AAE	Average Absolute Error
AiiDA	Automated interactive infrastructure and DAtabase

BIW	Body in White
CAD	Computer Aided Design
CAD0	Original part geometry which is designed in Computer Aided Design tool
CAD1	Non-ideal 'real' part geometry, containing all the imperfections resulting from all the stages of production process
CAE	Computer Aided Engineering
DFS	in-house company tool to create joining station set up and to generate easily LS-DYNA input file
EPFL	École polytechnique fédérale de Lausanne
FEM	Finite Element Method
GUI	Graphical User Interface
LHC	Latin Hypercube sampling
MAE	Maximum absolute error
MSE	Mean square error
OS	Operation System
RBF	Radial Base Function
STL	The Standard Template Library
SVR	Support Vector Regression
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Problem formulation

As the computational methods are developing further and further, each step of the production process is simulated to control the dimensional accuracy of assembled parts. Therefore, it is possible to test and choose the hardware at the early design stage, without real-life experiments. To identify the feasible process design a numerical optimization is usually performed. In reality parts used for the assembly contain specific for each part imperfections in the geometry caused by the manufacturing process. Numerical optimization in this case will result in hight computational costs, as the optimization has to be performed for each geometry separately. To investigate more efficient methods Data Mining techniques are studied. The method should be able to provide a realistic interpolation between non-linear simulation results. The problem of metamodele creation on top of simulated data is studied in this thesis. This metamodel should be able to catch the response of the assembly to the changing model parameters. Here the need for generation of a lot of simulation data arises.

In the automobile industry, during the production of Body in White (BIW) of automobile, especially in the production of hang-on parts such as doors (Figure 1.1), hoods and trunk lids, different joining techniques are used. Welding, clinching, roll

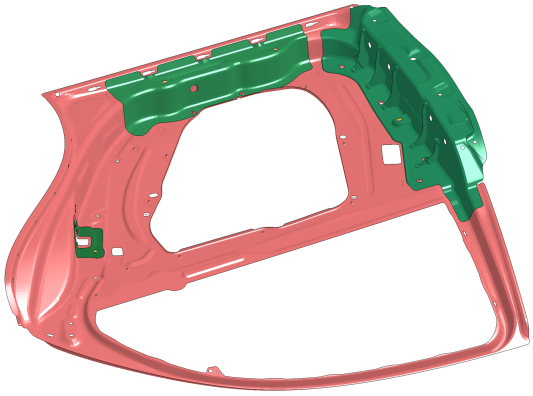


Figure 1.1: Body in White part example: door.

hemming and adhesive bonding are utilized to produce BIW assemblies. First the separate sheet metal parts are formed. Then the parts are brought together to the joining station, fixed in the right position by clamps and pins and joined together. Usually the joining processes are numerically simulated to ensure the geometrical accuracy of the produced parts. But parts resulted from the forming process are often deviated from ideal design due to tool wear, etc. These imperfections are usually corrected by clamp adjustment by experienced workers of the work shop.
The joining simulations of the BIW parts are the base for the presented thesis. Several tools and commercial software are combined for the comprehensive and easy use among employees of the company. The preprocessing tool developed in the company helps to create the simulation set-ups for the joining stations. In the end it writes the input files which are later used to run LS-DYNA jobs. Here the need of automated rearrangement of the set up arises in order to create the data sets for future Data Mining. In the present project, an independent tool is to be developed to satisfy specific company's needs and to combine existing tools.

1.2 Aim of the present work

In the presented project the framework is developed which is required to fulfill the functions to:

1. Create non-ideal input geometries with imperfections similar to the ones presented in the real parts
2. Create different workstation simulation cases
3. Run the simulation(s)
4. Extract the needed data in organized manner
5. Data Mining of the created/gathered/produced data

The framework is required to be:

1. Well-organized, with a clear communication structure between functions
2. Well-documented, functions and variables are explained

3. Accessible for future use and development

An automated environment for joining simulations, which would be able to create multiple simulation scenarios out of an initial case, simulate them in the LS-DYNA software and to extract and store the selected output for future Data Mining of the results.

1.3 Limitations

There are some limitations that had to be assumed in the project in order to simplify some of the tasks. These are:

1. The framework is built to function on top of certain tools: LS-DYNA and an in-house company tool

2. The framework's functionality is limited by computational cost and the time frame of the project

3. The deformations imposed by clamping are elastic

4. Clamping variation is reduced to locations only, such parameters as clamp size and amount of clamps are left for future analysis

5. The geometry variation is obtained by mapping function of the in-house tool

1.4 State-of-the-art

Industry 4.0 is a widespread subject in production engineering. It addresses the autonomy of the whole life-cycle of the product and the ability of the system to self-adjust [26]. Current industry, especially such fields as aeronautic and automotive, aims for the so called "Right First Time". The Digital Twin is introduced and intended to be a digital representation of the part during all cycles of production [28]. The ideal scenario would be to scan and simulate each part in each production stage to acquire the full control of the production process. The question here is the benefit as the required investment is huge. Other possibility is to control the assembly quality by simulating and analyzing only specific data set.

There are researchers who propose different methods to take into account the geometrical variations of the part into numerical simulations. With the help of mathematical tools it is possible to mimic the real-life surface error. Such approaches include second order shapes and Fourier series mesh morphing [27] and metric modal decomposition [14]. Nonlinear FEM analysis for forming process in combination with decomposition techniques to predict shape error modes [8].

Other studies tackle the problem of the taking into account the imperfections into the numerical simulations. To find the optimum set-up for an assembling station while varying the surface researchers use Monte Carlo simulation technique [7, 10, 12, 30]. As the variation of the parameters of the joining station result in non-linear nature of the problem, different approaches were proposed to reduce the computational time. For that purpose Dahlström and Lindkvis [7] developed the method of influence coefficient on top of the contact algorithm to capture the influence of geometry variation on the resulting assembly which resulted in a linear model. Andolfatto et al. [3] tested the use of Neural Networks to analyze the response of geometrical variation in the assemblies to pursue efficiency. The multi-input/multi-output nonlinear metamodel of the assembly was produced, while comparing different design parameters. There are multiple examples of metamodeling techniques applications in engineering and science. Zhu et al. [34] successfully implemented SVR metamodel for the vehicle crashworthiness design. In this study SVR models with different parameters were compared and the most accurate one was picked for further use in Optimization algorithm in order to reduce the mass and improve roof resistance force. Nik et al. [20] provided a comparative study of the metamodeling techniques for composite structures optimization. The fibers can be placed in different direction to adjust the mechanical properties of laminates. The metamodel provides the approximation model for the cases of compression and bending. The use of the metamodel in the optimization algorithm reduced the number of iterations 3 times comparing to pure optimization algorithm. There is also a biomechanical application present in the literature. Pan et al. [22] used Kernel Ridge Regression to predict soft tissue deformation after osteotomy. It was fitted using statistical information and data resulted from FEM simulations. The approximation model had an average prediction error of 0.91 mm, meaning that the model can produce an accurate enough predication of the operation outcome.

As of the platform implementations, which assist with automated simulations, there are

few examples present in the literature. Pizzi et al. [24] and Park and Dang [23] deal with several platform integration for the sake of the computational optimization. The first paper discusses an open source platform developed by a research group at EPFL, which is called AiiDA: automated interactive infrastructure and database [24]. The function of the software framework is concentrated around four subjects: Automation, Data, Environment and Sharing. The main idea is to automate the computational workflow with the possibility of parameter changes, to efficiently store the results and to share it with the whole research community if wished. The platform is mainly used in the scientific world for quantum-mechanical simulations. The second paper [23] presents the environment, which combines CAD/CAE integration together with Metamodeling techniques (Response Surface method and Radial Basis Functions) in order to solve the structural optimization problem.

Chapter 2

Theoretical framework

In this chapter Continuum Mechanics theory (see Sec. 2.1) and the corresponding Finite Element formulation (see Sec. 2.2) are covered to present the behavior of the model. The main points for the Contact algorithm reviewed in Sec. 2.3. Then the metamodeling techniques are discussed in Sec. 2.4 which would replace the numerical simulation and describe the response of the geometry by mathematical functions.

2.1 Governing equations

Continuum mechanics studies the nature of material and it's interaction with surroundings. To model the behavior of the matter the combination of general principles and constitutive equations are employed. The general principles are common for most of the materials and include real world physical laws, among which are the base one:

- Conservation of mass
- Balance of linear momentum
- Balance of angular momentum
- Conservation of energy
- Second law of thermodynamics

The meaning behind the conservation of mass principle is that the mass a body m stays the same along the timeline, while the volume $\mathbf{V}$ and density ρ may change. It is also

referred as continuity equation and can be written in spacial form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \, \mathbf{V}) = \mathbf{0} \tag{2.1.1}$$

Balance of linear momentum represents the equilibrium of the body: the sum of internal forces $\nabla \cdot \boldsymbol{\sigma}$ and body forces $\rho \, \mathbf{b}$ are equal to inertia forces $\rho \, \mathbf{a}$:

$$\nabla \cdot \boldsymbol{\sigma} + \rho \, \mathbf{b} = \rho \, \mathbf{a} \tag{2.1.2}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor. In reference configuration it takes the form:

$$\nabla \cdot \mathbf{P} + \rho_{\mathbf{0}} \, \mathbf{b} = \rho_{\mathbf{0}} \, \mathbf{a} \tag{2.1.3}$$

with $\mathbf{P}$ as the Piola-Kirchhoff stress tensor and ρ_0 as the density in the initial configuration. The balance of angular momentum outcomes in the symmetry of the Cauchy stress tensor:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^T \tag{2.1.4}$$

Another important relation is the Cauchy theorem which connects the Cauchy stress tensor and the traction force:

$$\mathbf{t} = \boldsymbol{\sigma} \cdot \mathbf{n} \tag{2.1.5}$$

On the other hand, constitutive equations represent the idealized material models, taking into account only certain relations, which could vary among different matters and loading cases. Examples of the material models are linearly elastic solids and linearly viscous fluids [16].

The widely employed and the simplest model is the linearly elastic homogeneous material model. The base for it is the elastic relation - Hooke's law, which provides the relation between stresses $\boldsymbol{\sigma}$ and strains $\boldsymbol{\varepsilon}$:

$$\boldsymbol{\sigma} = \mathbf{C} \colon \boldsymbol{\varepsilon} \tag{2.1.6}$$

here $\mathbf{C}$ is a stiffness tensor:

$$C_{ijkl} = \Lambda \delta_{ij} \delta_{kl} + \mu(\delta_{ik}\delta_{il} + \delta_{il}\delta_{jk}) \tag{2.1.7}$$

with Λ and μ as Lamé constants and δ_{ij} as Kronecker delta.

$$\delta_{ij} := \begin{cases} 1, & \text{for } i = j \\ 0, & \text{for } i \neq j \end{cases} \tag{2.1.8}$$

The Lamé constants are related with Young's modulus E and Poisson ratio ν:

$$\mu = \frac{E}{2(1+\nu)} \quad \text{and} \quad \Lambda = \frac{E}{(1+\nu)(1-2\nu)} \tag{2.1.9}$$

μ is also called shear modulus.

Derivation of the equations above can be found in [16] and [18] as well as transformation theorems [32].

2.2 FEM formulation

First the weak form is derived. The local form of the angular momentum in the current configuration is taken - Eq. (2.1.2). To satisfy the strong form for the approximate solution, the whole equation is multiplied by test function, which is $\boldsymbol{\omega} = \{\boldsymbol{\omega} | \boldsymbol{\omega} = 0 \, on \, \partial B_u\}$ and integrated over volume:

$$\int_B \boldsymbol{\omega} \cdot (\nabla \cdot \boldsymbol{\sigma}) \, dV + \int_B \rho \, (\mathbf{b} - \mathbf{a}) \cdot \boldsymbol{\omega} \, dV = 0 \tag{2.2.1}$$

Using the divergence theorem and integration by parts we acquire:

$$\int_B \boldsymbol{\sigma} : (\nabla \boldsymbol{\omega}) \, dV + \int_B \boldsymbol{\omega} \cdot \rho \, \mathbf{a} \, dV - \int_B \boldsymbol{\omega} \cdot \rho \, \mathbf{b} \, dV - \int_{\partial_t B} \boldsymbol{\omega} \cdot \mathbf{t} \, dA = 0 \tag{2.2.2}$$

The last term in the Eq. (2.2.2) represents the part of the boundary where the traction is applied. If $\boldsymbol{\omega}$ is rewritten as virtual displacement $\delta\boldsymbol{\varphi}$, the Eq. (2.2.2) represents the principle of virtual work with internal ($\delta\Pi_{int}$), kinematic ($\delta\Pi_{kin}$) and external ($\delta\Pi_{ext}$) terms:

$$\delta\Pi_{int} + \delta\Pi_{kin} - \delta\Pi_{ext} = 0 \tag{2.2.3}$$

To solve the equation the body is divided into n_e number of finite elements. The solution for any kind of problem can be presented in a form (from [33] Chap. 4.2):

$$\int_B (\dots)\, dV \approx \int_{B^h} (\dots)\, dV^h = \bigcup_{e=1}^{n_e} \int_{\Omega_e} (\dots)\, d\Omega = \bigcup_{e=1}^{n_e} \int_{\Omega_\square} (\dots)\, d\square \tag{2.2.4}$$

The first term here is the continuum description, the second term is the formulation in discretized domain. Third and forth terms in Eq. (2.2.4) present the assembly of all elements' contributions, but the last one is in the quadrature reference domain. The isoparametric mapping can be found in [33] Chap. 4.1.

For each element in the discretized domain the approximation is defined:

$$\mathbf{u_e} \approx \mathbf{u_e^h} = \sum_{A=1}^{n_n} \mathbf{N_A}\, \mathbf{u_A} = [\mathbf{N}]\{\mathbf{u}\} \tag{2.2.5}$$

where $\mathbf{u_e}$ is the displacement field for the element, exact solution, $\mathbf{u_e^h}$ is the displacement field, approximate solution, n_n is the number of nodes in the element, $\mathbf{N_A}$ is shape functions, used for the interpolation over the element and $\mathbf{u_A}$ is nodal variables of the displacement field. Last part in the Eq. (2.2.5) is shown in matrix notation, where:

$$[\mathbf{N}] = [N_1\mathbf{I}, N_2\mathbf{I}, \dots] = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \dots \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \dots \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \dots \end{bmatrix} \tag{2.2.6}$$

$$\{\mathbf{u}\}^T = [\mathbf{u_1}^T, \mathbf{u_2}^T, \dots] \tag{2.2.7}$$

with $\mathbf{I}$ as identity matrix. Same kind of form is used for the weighting function:

$$\boldsymbol{\omega}_e \approx \boldsymbol{\omega}_{\boldsymbol{e}}^{\boldsymbol{h}} = \sum_{A=1}^{n_n} \mathbf{N_A}\, \boldsymbol{\omega}_{\boldsymbol{A}} = [\mathbf{N}]\{\boldsymbol{\omega}\} \tag{2.2.8}$$

Similarly for acceleration $\mathbf{a}$:

$$\mathbf{a_e^h} = \sum_{A=1}^{n_n} \mathbf{N_A}\, \ddot{\mathbf{u}}_\mathbf{A} = \sum_{A=1}^{n_n} \mathbf{N_A}\, \mathbf{a_A} = [\mathbf{N}]\{\mathbf{a}\} \tag{2.2.9}$$

Substituting the derived approximations (2.2.5), (2.2.8) and (2.2.9) into the weak form

(2.2.2) we get the integrals for each element which is then assembled into the global matrix - the weighting coefficient ω is excluded as it is presented in each equation, and the equation must be satisfied for any ω:

$$\bigcup_{e=1}^{n_e} \left[\int_{\Omega_e} \mathbf{B^T} \boldsymbol{\sigma} \, d\Omega + \int_{\Omega_e} \rho \, \mathbf{N^T N a} \, d\Omega - \int_{\Omega_e} \rho \, \mathbf{N^T b} \, d\Omega - \int_{\Gamma_e} \mathbf{N^T t} \, d\Gamma \right] = 0 \tag{2.2.10}$$

where Γ_e is the part of the boundary where the traction is applied.

In the Eq. (2.2.10) $\mathbf{B}$ is a strain displacement matrix on the element level:

$$[\mathbf{B}] = [B_1, B_2, \ldots, B_n] \tag{2.2.11}$$

$$[\mathbf{B_i}] = \begin{bmatrix} \frac{\partial N_i}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial x_3} \\ 0 & \frac{\partial N_i}{\partial x_3} & \frac{\partial N_i}{\partial x_2} \\ \frac{\partial N_i}{\partial x_3} & 0 & \frac{\partial N_i}{\partial x_1} \\ \frac{\partial N_i}{\partial x_2} & \frac{\partial N_i}{\partial x_1} & 0 \end{bmatrix} \tag{2.2.12}$$

In the Eq. (2.2.10) $\int_{\Omega_e} \rho \, \mathbf{N^T N} \, d\Omega$ represents the mass matrix and has the form:

$$[\mathbf{M}] = \int_{\Omega_e} \rho \, \mathbf{N^T N} \, d\Omega = \int_{\Omega_e} \rho \begin{bmatrix} N_1 N_1 & N_1 N_2 & \ldots & N_1 N_n \\ N_2 N_1 & N_2 N_2 & \ldots & N_2 N_n \\ \ldots & \ldots & \ldots & \ldots \\ N_n N_1 & N_n N_2 & \ldots & N_n N_n \end{bmatrix} d\Omega \tag{2.2.13}$$

After all the contributions for each element are summed, the global matrices are built. Then the equation can be represented in terms of forces: the first term on the left is the internal force, the second one is the kinetic force and the term on the right is the external force:

$$\mathbf{f_{int}} + \mathbf{f_{kin}} = \mathbf{f_{ext}} \quad \text{or} \quad \mathbf{f_{int}} + \mathbf{Ma} = \mathbf{f_{ext}} \tag{2.2.14}$$

If to solve the problem for the static case the kinematic term can be neglected. The problem takes the form:

$$\mathbf{f(u)} = \mathbf{f_{int}} - \mathbf{f_{ext}} = 0 \tag{2.2.15}$$

The Eq. (2.2.15) can be solved using Taylor expansion:

$$\mathbf{u_{k+1}} = \mathbf{u_k} + \mathbf{\Delta u_{k+1}} \tag{2.2.16}$$

Leading to:

$$\mathbf{f}(\mathbf{u_{k+1}}) = \mathbf{f}(\mathbf{u_k}) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \mathbf{\Delta u_{k+1}} = 0 \quad \Rightarrow \quad \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \mathbf{\Delta u_{k+1}} = -\mathbf{f}(\mathbf{u_k}) \tag{2.2.17}$$

In $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$ the u_k is held fixed. This term is also called stiffness matrix $\mathbf{K}$:

$$\mathbf{K} := \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \mathbf{K_{int}} - \mathbf{K_{ext}} \tag{2.2.18}$$

Now that the problem is linearized, it can be solved using different iterative schemes. One of them is Newton-Raphson scheme, it solves the problem in steps:

1. $\mathbf{u_0}$ is used as a starting guess and $k = k + 1$ is taken
2. $\mathbf{K \Delta u_{k+1}} = -\mathbf{f}(\mathbf{u_k})$ is solved for $\mathbf{\Delta u_{k+1}}$
3. Update $\mathbf{u_{k+1}} = \mathbf{u_k} + \mathbf{\Delta u_{k+1}}$ is done
4. Convergence is checked by $\mathbf{\Delta u_{k+1}}$ reaching the predefined tolerance value

If the kinematic term is involved, the problems becomes time dependent. Then the problem is additionally iterated over time. Explicit and implicit integration techniques are then employed. The explicit method solves the problem using the values acquired at the previous time step. The scheme requires small step size to produce a stable solution, which is reasonable for high frequency problems such as car crash simulation. As for the implicit method - the solution depends on quantities in both previous and current time step, leading to the nonlinear system to be solved at each time step. That integration scheme is mostly used for low frequency problems. See Chap. 6 in [33].

2.3 Contact formulation

For the contact problem the problem is formulated for two bodies and contains additional traction term resulted from the contact of the bodies. It can be represented in terms of

virtual work with contribution from contact ($\delta\Pi_{cont}$):

$$\sum_{I=1}^{2}\left[\delta\Pi_{int,I}+\delta\Pi_{kin,I}-\delta\Pi_{ext,I}-\delta\Pi_{cont,I}\right]=0 \tag{2.3.1}$$

The weak form is then:

$$\begin{aligned}\sum_{I=1}^{2}\Bigg[&\int_{B_I}\boldsymbol{\sigma_I}\colon(\nabla\boldsymbol{\omega_I})\,dV_I+\int_{B_I}\boldsymbol{\omega_I}\cdot\rho_I\,\mathbf{a_I}\,dV_I\\&-\int_{B_I}\boldsymbol{\omega_I}\cdot\rho_I\,\mathbf{b_I}\,dV_I-\int_{\partial_t B_I}\boldsymbol{\omega_I}\cdot\mathbf{t_I}\,dA_I-\int_{\partial_C B_I}\boldsymbol{\omega_I}\cdot\mathbf{t_{CI}}\,dA_I\Bigg]=0\end{aligned} \tag{2.3.2}$$

where $\partial_C B_I$ is a part of the boundary where the contact of the bodies takes place. The contact traction is decomposed in two parts - normal $\mathbf{t_N}$ and tangential $\mathbf{t_T}$ contact forces with normal basis vector $\mathbf{n}$ and two tangential basis vectors $\mathbf{a_1}$ and $\mathbf{a_2}$:

$$\mathbf{t_C}=\mathbf{t_N}+\mathbf{t_T}=t_N\mathbf{n}+t_T^1\mathbf{a_1}+t_T^2\mathbf{a_2} \tag{2.3.3}$$

Another important variable is the contact gap $\mathbf{g_C}$, which is decomposed in the same way:

$$\mathbf{g_C}=\mathbf{g_N}+\mathbf{g_T}=g_N\mathbf{n}+g_T^1\mathbf{a_1}+g_T^2\mathbf{a_2} \tag{2.3.4}$$

The contact condition are posed as an optimization problem. For the normal contact the Kuhn-Tucker conditions must be satisfied:

$$\begin{aligned}&t_N\leq 0\\&g_N\geq 0\\&t_N\,g_N=0\end{aligned} \tag{2.3.5}$$

here t_N is a traction force resulting from normal contact and g_N is a normal gap between the contact bodies. Physically it means that either there is a distance between two bodies or the contact force. There are several methods to enforce the condition (2.3.5) in the problem: Lagrange multiplier, Penalty, Barrier, Nitsche, Augmented Lagrange methods (see Chap. 6.3 in [32]). As an example, Penalty method assumes a spring with high value spring constant to guarantee impenetrability. The normal contact pressure is then

represented with spring constant k and contact penetration $d_N = -g_N$:

$$\mathbf{t_N} = k d_N \mathbf{n} \tag{2.3.6}$$

In case if the friction is omitted, it results in the contact contribution in the weak form:

$$\int_{\partial_C B} \boldsymbol{\omega} \cdot \mathbf{t_C}\, dA = \int_{\partial_C B} \boldsymbol{\omega} \cdot (k d_N \mathbf{n})\, dA \tag{2.3.7}$$

The problem is then discretized in the way shown in the previous Chap. 2.2. What is distinct for the contact algorithm is that it contains an additional loop for each iteration where the surfaces are checked for penetration.

To provide the efficient and robust consideration of contact the Mortar segment-to-segment method is used, which penalizes the surface in contact not on the node scale, but the whole volume along the contact area is forced to be zero [25].

The derivation of contact algorithm presented in the current section as well as more details can be found in [32].

2.4 Data Mining - Metamodelling techniques

Common way to reduce the computational costs and to extract the information from the simulated model is to construct a metamodel. The metamodel describes the behavior of the given model as a mathematical function $f(\mathbf{x}) = \mathbf{y}$, where $\mathbf{x}$ is an input set of variables and $\mathbf{y}$ is an output response of the model to these variables. A fitting set $(\mathbf{x_i}, \mathbf{y_i})_{i \in \{1 \dots N\}}$ with N as a sampling size is used to create a metamodel. To construct a metamodel f^* of the given problem the error ε should be minimized:

$$\varepsilon = \sum_{i=1}^{N} |f^*(\mathbf{x_i}) - \mathbf{y_i}| \tag{2.4.1}$$

which results in a good approximation of the data which was not used for fitting [3].

The base for all the metamodeling techniques is the linear regression. There are various studies presenting different metamodeling techniques used and compared for the simulation simplifications. Andolfatto et al. [3] used Artificial Neural Networks to study non-linear effects of the geometry variation on the simulation results of the sheet metal

assembly, which is similar to the topic of the present study. Other researchers applied metamodeling techniques for various problems, such as a Decision Support Systems together with optimization technique [17], an engineering optimization problem [4], a vehicle crashworthiness design simulation [34] and a stiffness analysis of the composites [20]. A brief comparison of the methods are presented below on the base of studies cited above. The metamodeling techniques are:

- *Polynomial regressions* are said to result in good approximation on the local scale, but do not return a good prediction for highly nonlinear cases.
- *Regression splines* are suitable only for univariate or bivariate type of problem.
- *Kriging or Spatial correlation* is computationally expensive for the problems with many design variables, as the maximization problem has to be solved.
- *Radial basis function (RBF)* construct the metamodel in terms of basis functions by taking into account Euclidean distances. it is used for multivariate data.
- *Artificial Neural Networks* are able to create multi-input/multi-output metamodels, but require network architecture to be taken care of.
- *Kernel regressions* return good approximation with lower computational costs, such regressions include Support Vector Regression (SVR) and Kernel Ridge Regression.

For the project Kernel regressions are tested due their computational efficiency and availability of the algorithm in Python. Both SVR and Ridge Kernel Regression use kernel trick to map the non-linear problem into linear. As an example, Wang et al. [31] successfully implemented SVR based optimization for the highly nonlinear engineering problem - sheet forming, Pan et al. [22] constructed a promising biomechanical metamodel of tissue deformations using Kernel Ridge Regression basing on both FEM and statistical data.

2.4.1 Kernel trick

Kernel trick is performed by using kernel function. It is defined to be a real-valued function of two argument : $\kappa(\mathbf{x}, \mathbf{x}')$, which is typically symmetric and non-negative [19]. The main idea behind the kernel is to assign the weights to $\mathbf{x}$ when it is located in neighborhood of $\mathbf{x}'$, therefor measuring the similarities between the input objects [13]. Such functions include

as an example linear, polynomial and Radial Basis functions. As there is nonlinearity presented in the present study case and for the reason mentioned in Sec. 2.4, RBF was chosen to be used. The RBF, or specifically squared exponential kernel (SE kernel) or Gaussian kernel, has the form:

$$\kappa(\mathbf{x}, \mathbf{x}') = exp\Big(- \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\Big) \tag{2.4.2}$$

Here σ^2 is bandwidth, which controls the width of the neighborhood [13].

2.4.2 Support Vector Regression

Support Vector Regression is based on Support Vector Machines, which are used for statistical classification problems. To build the regression $f^*(\mathbf{x})$ not all of the data points are used. There is the specific value ϵ which creates the epsilon tube in the region of regression. It results in the condition, that the error is neglected if it is less than ϵ. Starting from the linear regression model:

$$f^*(\mathbf{x_i}) = \mathbf{w}^T\mathbf{x_i} + \mathbf{w_0} \tag{2.4.3}$$

where $\mathbf{w}$ is weighting vector term and $\mathbf{w_0}$ is bias. The optimization problem takes form [19]:

$$J = C\sum_{i=1}^{N} L(\mathbf{y_i}, f^*(\mathbf{x_i})) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.4.4}$$

where C is the regularization constant. The Eq. 2.4.4 ensures the reduction of the error of the metamodel (loss term) and flatness of the regression (regularization term). Epsilon insensitive loss function is used:

$$L_\epsilon(\mathbf{y_i}, f^*(\mathbf{x_i})) := \begin{cases} 0, & \text{if } |\mathbf{y_i} - f^*(\mathbf{x_i})| < \epsilon \\ |\mathbf{y_i} - f^*(\mathbf{x_i})| - \epsilon, & \text{otherwise} \end{cases} \tag{2.4.5}$$

this equation results in epsilon tube. The components that result from the non-vanishing terms are called *Support Vectors*. After some manipulation ([29]) one can see that the

solution of the optimization problem takes form:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_\mathbf{i} \mathbf{x_i} \tag{2.4.6}$$

After plugging Eq. 2.4.6 into Eq. 2.4.3 we get:

$$f^*(\mathbf{x}) = \sum_{i=1}^{N} \alpha_\mathbf{i} \mathbf{x_i}^T \mathbf{x} + \mathbf{w_0} \tag{2.4.7}$$

Here the product $\mathbf{x_i}^T\mathbf{x}$ can be replaced by kernel function $\kappa(\mathbf{x_i}, \mathbf{x})$ resulting in kernalized solution:

$$f^*(\mathbf{x}) = \sum_{i=1}^{N} \alpha_\mathbf{i} \kappa(\mathbf{x_i}, \mathbf{x}) + \mathbf{w_0} \tag{2.4.8}$$

For more detailed overview of SVR see [29], as well as example solutions [31],[34].

2.4.3 Kernel Ridge Regression

Ridge regression has a linear model:

$$f^*(\mathbf{x_i}) = \mathbf{w}^T \mathbf{x_i} \tag{2.4.9}$$

with the optimization problem to solve, similar to SVR:

$$J = \sum_{i=1}^{N} (\mathbf{x_i w} - \mathbf{y_i})^2 + \lambda \|\mathbf{w}\|^2 \tag{2.4.10}$$

Here λ is a shrinkage parameter, which is a small positive value used as a regularization constant to manage the overfitting of the model. Comparing to the SVR: $\lambda = 1/C$. After some manipulations (see Chap. 14 [19]) the solution can be represented in the same form as 2.4.6, except the α_i is different. It results in the kernalized model:

$$f^*(\mathbf{x}) = \sum_{i=1}^{N} \alpha_\mathbf{i} \kappa(\mathbf{x_i}, \mathbf{x}) \tag{2.4.11}$$

For more details on the derivation see Chap. 14 of [19] and example [22].

2.4.4 Sampling

To produce a good metamodel quality data should be sampled. In the current study two sampling techniques were tested: random uniform and Latin Hypercube sampling techniques. For the first one the algorithm does not check the samples created before and after, which might result in repeated data sets or non-uniform distribution of the data. To reduce the amount of fitting data and therefor computational times more clever sampling methods are needed, where the sampling is controlled.

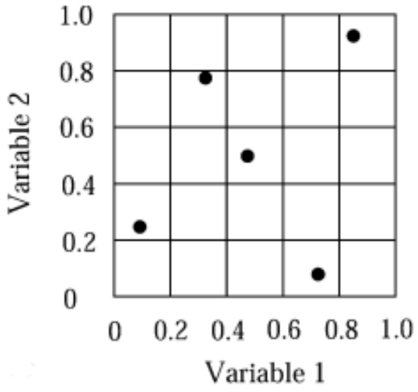


Figure 2.1: Latin Hypercube sampling for two variables (Figure by Olsson et al. [21]).

Several articles concentrated on metamodeling utilizing Latin Hypercube sampling technique [9], [17], [4], [15], [20]. It results in a uniform coverage of the sampling space in the selected range [17]. The main idea behind it is to take each value of each parameter only once, like it is shown on the Figure 2.1. For more information about the LHC sampling technique see [21].

Chapter 3

Software framework for automated simulations

The expected function of the software framework is to provide an automated environment, which would be able to explore the joining station. It is achieved by taking into account the geometrical variation of the parts, which occurs during the manufacturing process and influence of the parameters of the joining station onto the resulted geometry of the assembly. The numerical simulation is utilized to solve the sampling cases resulted from the variation. In the end all the needed data is extracted and the metamodel is fitted.
This Chapter starts with the Sec. 3.1, which discusses the tools utilized in the framework to provide the functionality. Then joining station setup and computational aspects are briefly introduced in Sec. 3.2. Sec. 3.3 provides details on the structure of the framework.

3.1 Tools

3.1.1 Python

The Python programming language is used for coding the computational framework due to its' functionality and comprehensibility, e.g. the interface for data mining functionality is already well defined. It is an object-oriented language which is well-structured, in other words it supports functional programming. Python is compatible with most of the systems and platforms (both Windows and Linux), as well as capable to interpret many file extensions. It has a wide library for text parsing, data processing, plotting,

data mining and passing the processes through [11]. Therefore Python makes it possible to built a functional framework satisfying all the requirements. The list with standard modules which were used in the implementation of the framework can be found in App. A.

3.1.2 LS-DYNA

LS-DYNA is a commercial finite element software, which employs multiple material models to solve real world problem. The code can be adjusted to different applications for various fields, such as automobile crash analysis, manufacturing process of sheet metal and other physical events. LS-DYNA is command line driven, the input for it is a single file to execute. Sum of all the qualities mentioned above make it suitable for contact problem, as well as for an automation. Therefore it can be successfully implemented in the framework of the project.[6]

3.1.3 DFS tool

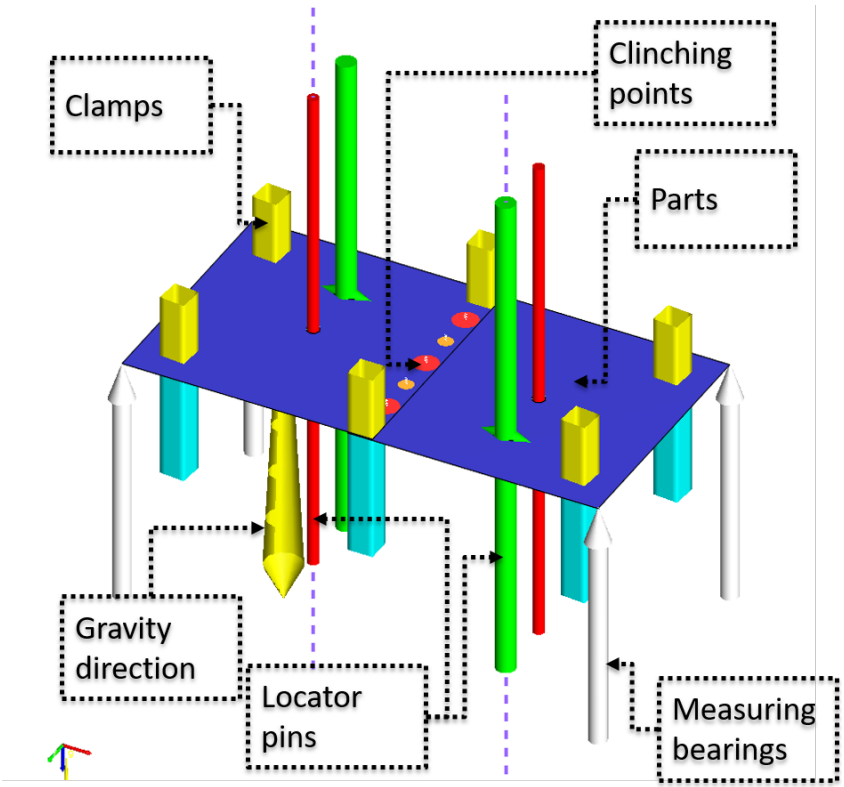


Figure 3.1: Joining station setup in the in-house tool (DFS).

The setup of the joining station and the whole simulation is prepared with the help of in-house tool called "DFS". The parts' geometries are imported as a mesh, which was prepared and located beforehand in the mesh handling program. The DFS tool has several joining options as modules: clinching, roll hemming, welding etc. The tool assists in defining locator pins, clamps, joining tool and measuring bearings (see Fig. 3.1). In the presented project the clinching process was tested as an example. The simulation parameters are also defined in the tool. As an output the DFS tool creates input files for LS-DYNA (key files), which contain key files for all stages of the joining:

1. placing the parts in specified order onto lower clamp
2. clamping of the part
3. clinching
4. placing on measuring bearings taking into account gravity
5. removing gravity

The DFS project file used by tool itself is an xml type of file, which contains both editable and non-editable (binary) data. The modifiable part gives access to the parameters, such as clamping location, size, direction and most of the other settings which are defined in the GUI of the tool. The binary part consists mainly of the part geometries. The DFS tool can be invoked by the command line, which makes it scriptable. With this tool it is possible to regenerate clamps, deform the geometry by mapping the mesh onto cylinder surface, save new project file and write LS-DYNA input file.

3.2 Joining station setup and computational aspects

The prototype for the test is a flat 200x200 mm sheet aluminum part with 1 mm thickness (See Table 3.1). The elastic linear model is used for the material of the part (See Sec. 2.1). Shell elements are used with maximum element size is 10 mm - see Fig. 3.2. There are also two locating holes in the part for locator pins. Two of such parts are

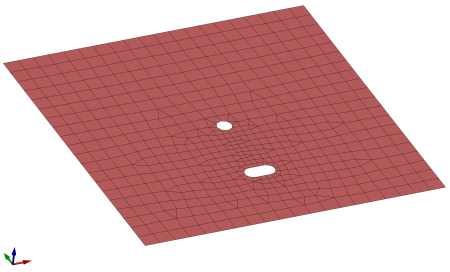


Figure 3.2: Part.

clinched together to result in the joined assembly - see Fig. 3.1. The parts are held in place with 6 clamps. For each part two locator pins are used to locate parts correctly in space.

For all contacts involved in the case the implicit segment-to-segment Mortar penalty contact algorithm is used (for general overview see Chap. 2 or for more detailed explanation [32], [33] and [1]). For both clamps and clinches the rigid body type of material is used (See Table 3.1 for properties).

The joining simulation contains 5 jobs as it was described in Sec. 3.1.3. First the part is placed onto the lower clamps until the equilibrium is established. In the second job the clamps are closed. For that they are first set 20 mm away from the part and then moved back, until the predefined position is reached. The positions defined in DFS tool are the locations for the closed state of clamps. During this step the contact with sheet metal is established and the parts are deformed elastically. Third procedure is clinching. The clinches are closed, deforming the parts (forming contact in LS-DYNA). To imitate the clinching a beam element is connected to both parts in the region of clinching (For beam properties see Table 3.1, Connection). The beam is predefined and already connected to one part. With the spotweld type of contact in LS-DYNA it ensures the connection to the second part. In the fourth job the assembly is placed onto the bearing to check the deformation under the gravity. The final step is to remove the gravity. In the presented project the assembly without the gravity influence is explored.

Materials				
Material	Model	Density, ρ, $[g/mm^3]$	Young's modulus, E, $[GPa]$	Poisson ratio, ν
Aluminum	elastic	2.4000E-6	70	0.28
Rigid body	rigid	7.8300E-6	210	0.3
Connection	elastic	7.8300E-6	40	0.3

Table 3.1: Materials used.

Each job was solved on Linux using about 16GB of memory. That resulted in about 20 to 120 minutes of computation per one job. The time was varying depending on how hard it was to establish the equilibrium in the joining station during clamping and joining. As an example, parts with larger nonuniform deformation along the part, as well as non-balancing clamping result in longer computational times or non-convergence.

3.3 Input-Output handling

The general structure of the framework can be divided in 3 parts:

- create
- solve
- analyze

Each step of the framework is called separately for the reason of flexibility, as well as requirements: `create` is called on Windows OS, as DFS tool is only adapted for it, while `solve` is run on Linux, because it is number-crunching machine of the project. The flow chart for the framework can be seen on Fig. 3.4. Case creation contains two loops, one of which is nested. First the 'real' imperfect geometry case is randomized. For this geometry a specified quantity of clamping cases are randomly created. The procedure is repeated by number of geometry cases requested by the user. The cases are solved with the help of LS-DYNA. In the end the metamodel is built using the converged results. The detailed description of each part of the framework and Data structure can be found in following chapters.

The Input/Output data flow can be seen on Fig. 3.3. The only input file for the framework is the `project_file.ACPROX`. During `create` routine input file for LS-DYNA to each case is created with the help of DFS tool. It contains the job information, along with parts' geometry and station set up. During `solve` routine each job is solved and result files are produced. The first file contains resulted from the assembly geometry in a mesh format and the second one is the resulted stresses. It was decided to save and extract certain files for the future data - See Fig. 3.3 and App. C for more detailed description.

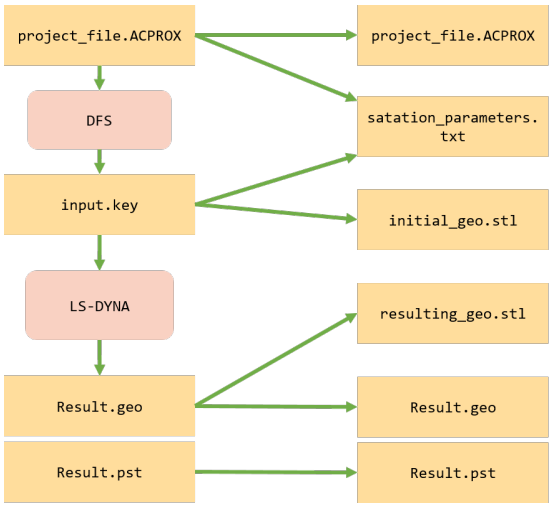
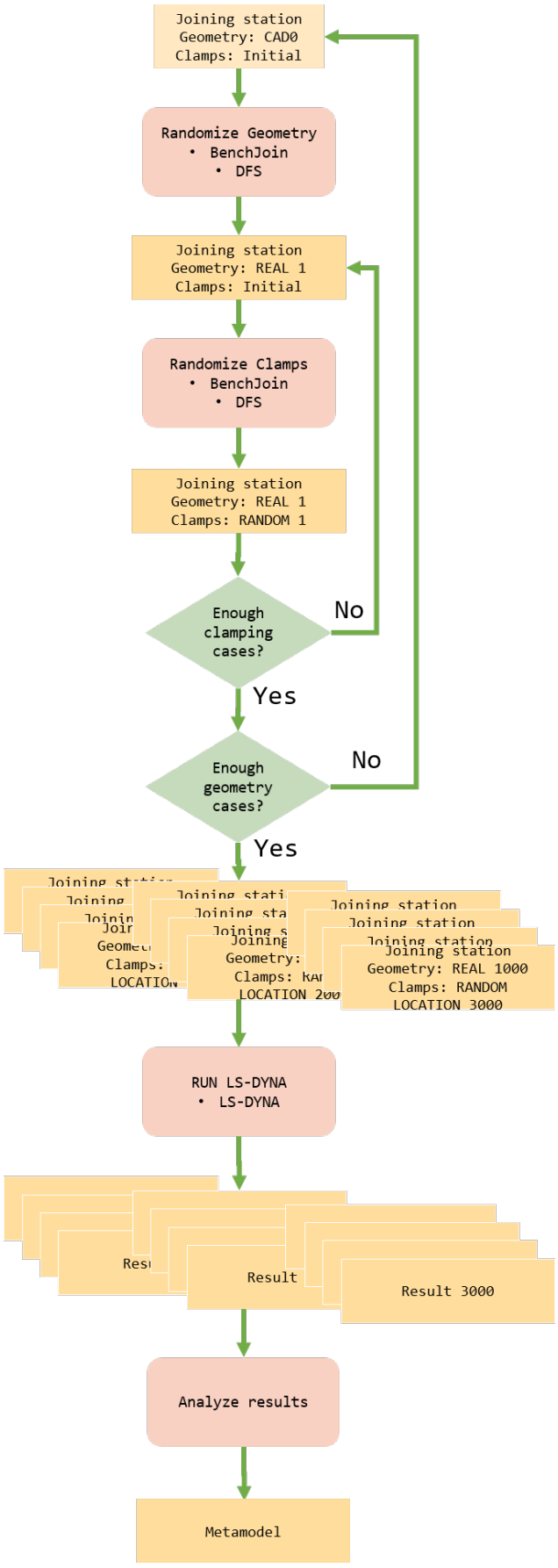


Figure 3.3: Data Flow.

Figure 3.4: Flow chart describing automated simulation environment.



3.3.1 Geometry and clamping variation

As it will be explained in Sec. 3.3.2, first the geometries are randomized. Mesh of each part is mapped onto a cylinder (See Fig. 3.5a) using DFS command line. Few parameters are specified to produce the cylinder. All the ranges for both geometry and clamping are set in `VARIATION_RANGES` dictionary (App. B). The ranges for geometries are:

- translations from the center in x and y (Fig. 3.5, two straight arrows)
- rotation angle Fig. 3.5
- maximum deviation from the ideal geometry - Fig. 3.5
- side of the part (above or below)

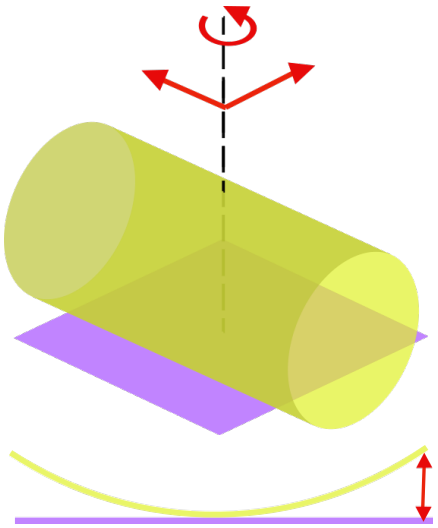


Figure 3.5: Cylinder for geometry mesh mapping.

For moving the clamps several values are specified:

- location in x, y and z; randomized in specified range delta is added to initial clamp location. The resulted positions can be seen as a point inside the cube - See Fig. 3.6, red cube. After the position is specified, the DFS tool moves the clamps to the closest point on the body, so that the clamp lies on the same plane with the body - Fig. 3.6 (1)-(3).
- shimsing provides a function to move the clamp from the body. It is moved by the value and direction. In this project the clamp was moved in the closing clamp direction, which is coincidental with z direction - Fig. 3.6 (4).

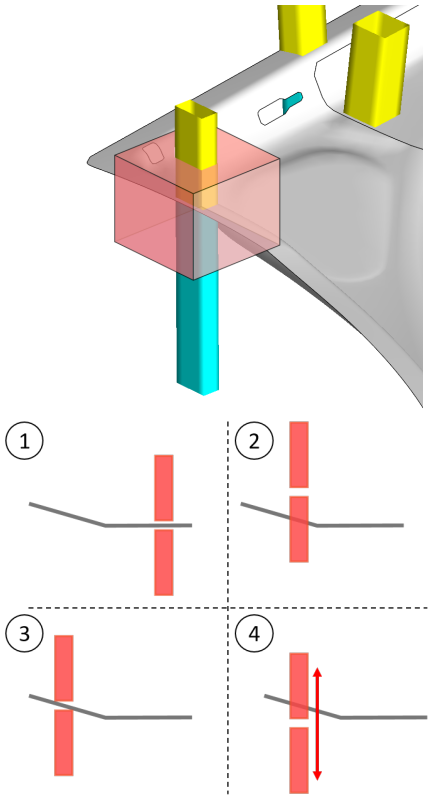


Figure 3.6: Clamping variation.

As a result the true new location of the clamp is not
specified in the project file, but can be extracted from the mesh file of the geometry key file for the LS-DYNA. See Fig. 3.8 for more details on the `extract_clamp_locations` function.
There are additional parameters that could be varied (clamp size), which are also implemented in the code for the future use.

3.3.2 Create cases structure

The data structure of `create_cases` function is presented in Fig. 3.7. The input for the software framework is the `ACPROX` file with `CAD0` state and initial clamping locations. While `CAD0` state is the default one and can be easily set to it by flag `CAD0`, the deformed state is optional and can be activated through batch command only if it is present in the initial station project file. The bending states are applied on surfaces with arbitrary values and set active beforehand with DFS tool, so that the names of the states can be parsed from the project file and set active later on when needed (The name of the bending state is used in the batch command). The initial project file is parsed with `parse_xml` function, so that the `root_xml` is in the memory and any needed parameter can be accessed by using tags. That is done with `get_specification` command and all needed parameters (`specs`) are kept in memory for future use and modification. Before the randomization is started, the base case is created using the unchanged geometry with flag set to `CAD0` and initial clamping. This case is required for future analysis and comparison of the created cases with ideal geometry. To produce new cases first the DFS tool is activated and requested by command line to deform the certain part by randomized `bending_parameters`. Only one body at a time can be deformed which leads to multiple calls of the tool and reduces the performance. That is achieved by command `run_dfs_bend` and it results in new updated project file with `CAD1` geometry. For this geometry the new sets of variables are created with function `randomize_specification` by adjusting the saved in the memory parameters `specs`. New specification are then written into the new project file using `write_specification` function. This project file is then run with DFS tool with `run_dfs_create_case`, which sends a command to the tool to regenerate the clamps and then write the case folder which contains the LS-DYNA jobs. As it was mentioned earlier the true locations of the clamps can only be received

from LS-DYNA key files. Therefore the locations of the clamps are extracted at this point, by checking the node locations and element connectivity of each clamp (See Fig. 3.8). The center of the clamp along with normal direction and size of the clamps are then stored in `station_parameters.txt`.

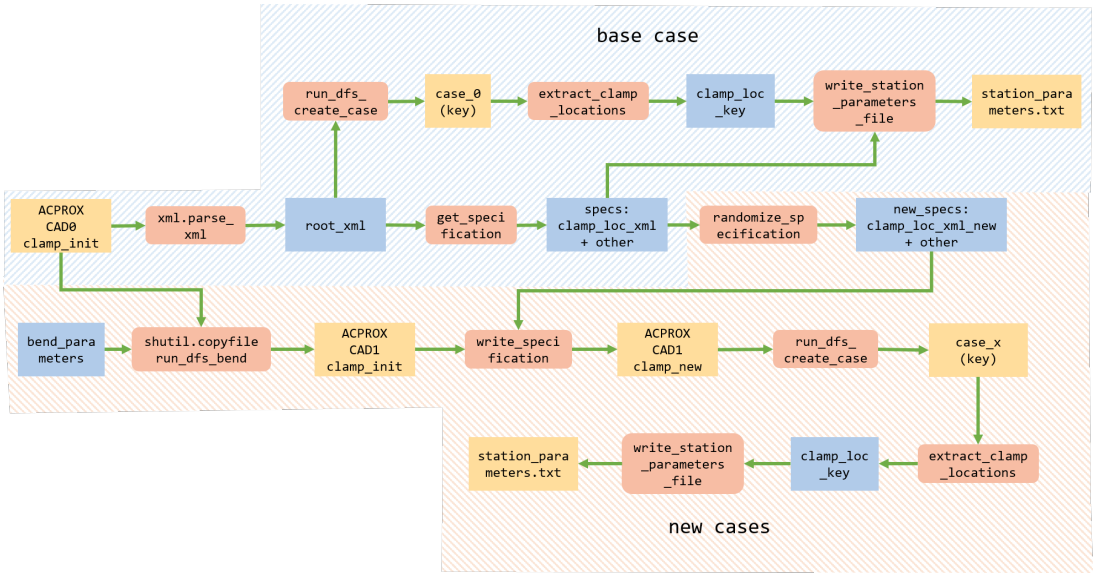


Figure 3.7: Data structure of the `create_cases` function.

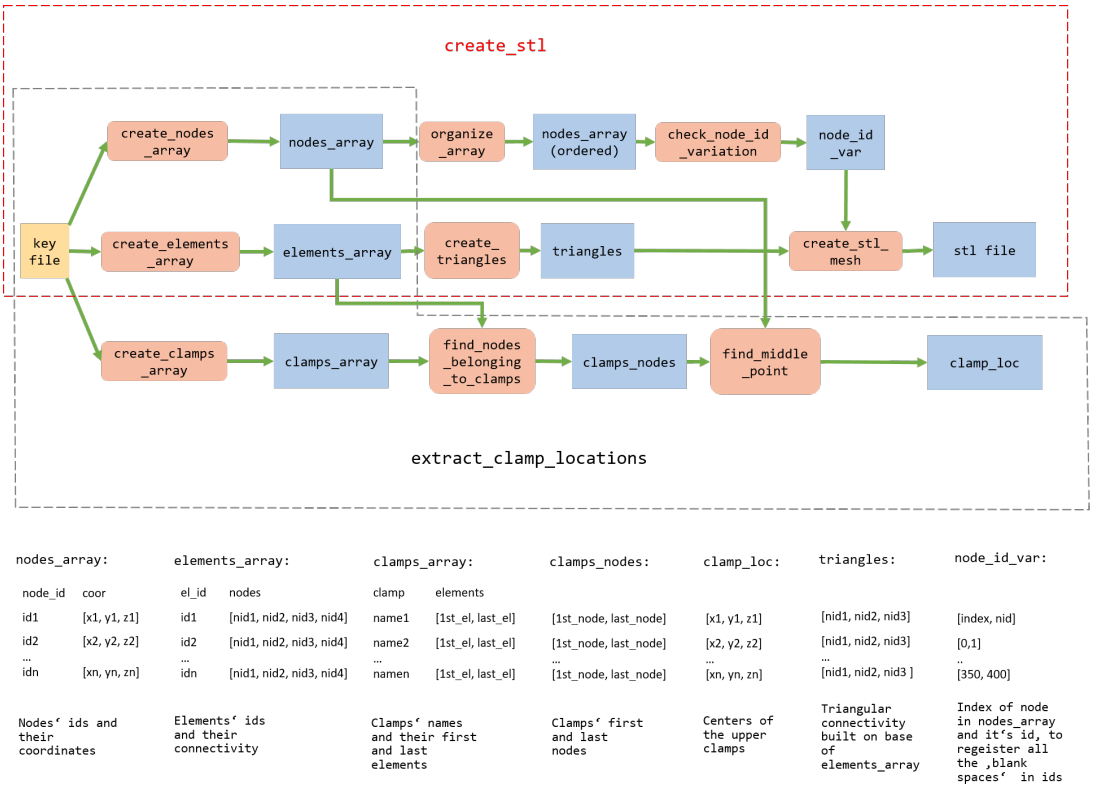


Figure 3.8: Data structure of the `create_stl` and `extract_clamp_locations` functions, common functions used.

3.3.3 Solve cases structure

During the `solve` routine each of the cases resulted from `create` module are solved one by one. The flowchart is presented on Fig. 3.9. The key files are passed to the LS-DYNA for solving and if the operation is successful the geometries are saved in the STL format. To do that the geometry is the node locations and connectivity is extracted from the key file and triangulated. See Fig. 3.8 for more details on the `create_stl` function. After that all the files (see Fig. 3.3) for the database are moved to the separate folder and the job folder with the key files and results are deleted, as they usually occupy a lot of space. The same procedure is done when the job does not converge.

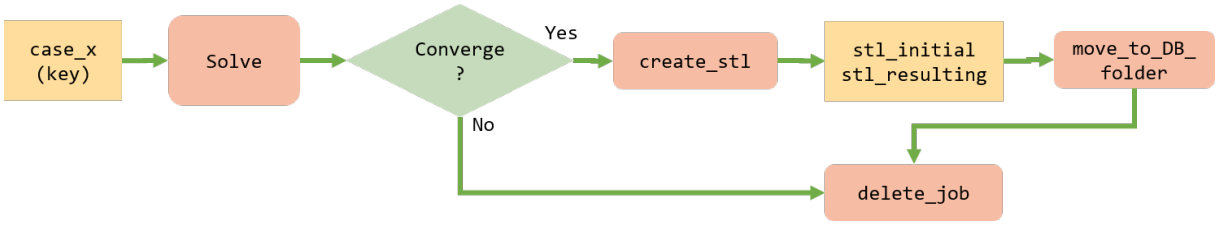


Figure 3.9: Solve flowchart, for one case.

3.3.4 Analyze cases structure

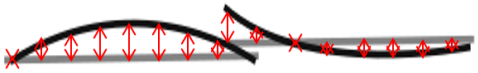


Figure 3.10: Mean deviation.

To build the regression all the variables and the response of the model should be extracted. All the variables are stored in the `station_parameters.txt` file. As for the measurement of the quality of the assembly one value had to be selected. The mean of all the point deviations along the geometry between `CAD0` and `CAD1` was computed (Fig. 3.10). To compare it to the maximum deviation, it represents the behavior of the model along the whole part, not only local effects. Other solutions should be checked in the future, as it might not be accurate for the uneven, unequal mesh.

The flowchart for the `analyze` module is shown on Fig. 3.11. First all the information is parsed from files. Each resulted assembly is compared to `CAD0` and the mean for it is calculated. The regression is built using the clamp locations as parameters and mean deviation as a model response. The testing data is separated from the fitting data for the purpose of checking the regression.

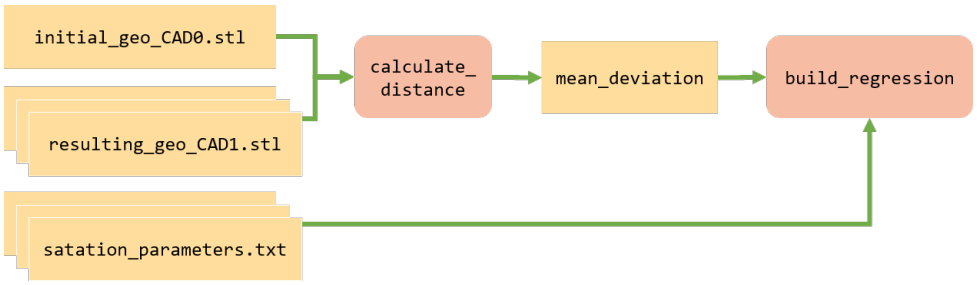


Figure 3.11: Analyze flowchart.

Chapter 4

Results

4.1 Testing set up

A metamodel was tested for the simplified cases of one and two varying parameters as the time to perform the computation is restricted. For the both of the tests the geometries were varied to create the imperfections. Same imperfect geometry was used for both of the tests - see Fig. 4.1. The interpolation between the geometries was excluded from the design variables, as it enlarges the design dimensionality significantly.

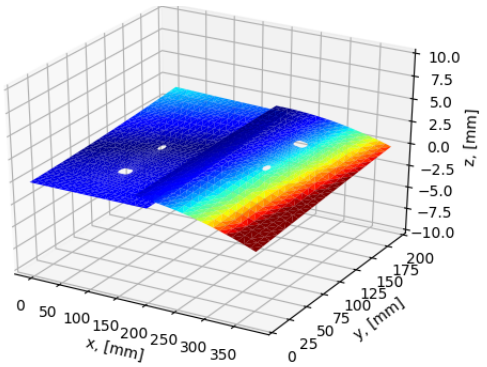


Figure 4.1: Geometry with imperfection.

Before the model is fitted the input variables are preprocessed. Each variable is centered and scaled to be in the range $[-1, 1]$. That is required by algorithm [2] to provide the stable solution.

As it was explained in Sec. 2.4, there are parameters which help to adjust the metamodel quality. For SVR it is penalty error C and epsilon tube ϵ, for Ridge regression it is α, where $\alpha = (2C)^{-1}$ [2] (same meaning as in SVR, but not the same number in practice as it is seen in the later testing). Cherkassky and Ma [5] performed the investigation of the parameter selection for SVR. According to it, $C = max(|\bar{y} + 3s_y|, |\bar{y} - 3s_y|)$ with $\bar{y}$ and s_y as mean and standard deviation of the the training response. The same study suggests that epsilon calculation requires a separate model to estimate the noise. It was

decided to check the values by testing and error assessment. The reason for it is that the recommended penalty parameter did not return good prediction and the epsilon tube calculation required new model implementation.

For both of the metamodels RBF kernels were chosen, as it proved to return the best approximation for nonlinear metamodels (See Sec.2.4.1, [5]).The RBF width parameter σ for eq. 2.4.2 was chosen according to [5].

To check the built regressions about 10-25 percent of the simulated data was excluded. That data should be not coincident with fitting data to provide true assessment. For evaluation of regressions 3 measures were selected similar to [17]:

- Mean square error:
$$MSE = \frac{1}{n}\sum_{i=n}^{n}(y_i - y_i^*)^2 \tag{4.1.1}$$
- Average absolute error
$$AAE = \frac{1}{n}\sum_{i=n}^{n}|y_i - y_i^*| \tag{4.1.2}$$
- Maximum absolute error
$$MAE = max|y_i - y_i^*| \tag{4.1.3}$$

here n is number of test samples, y_i resulted from the simulation outputs and y_i^* are predicted outputs. The coefficient of determination R^2 of the prediction was also checked to estimate the quality of the fitted model:

$$R^2 = (1 - \frac{u}{v}), \tag{4.1.4}$$

with u as a residual sum of squares and v a total sum of squares [2]. R^2 shows the quality of the prediction of the model, it is equal to 1 if the regression is able to predict exactly all the data used for fitting.

4.2 One design variable metamodel

For the first investigation clamp number 5 was varied only in z direction (shimsing), which is normal to the flat part (Fig. 4.2). The clamp was chosen arbitrary. The range for the variation was decided to be from -5 mm to 5 mm from initial location with step of 0.5 mm.

It resulted in 23 cases, 6 of them were excluded from fitting for the testing of the regression. Different parameters were used to build the regression, starting from the default-/near default values specified in the algorithm documentation [2]. The comparison of the different parameters can be seen in App. D. The chosen parameters which were used for the building of the regressions are presented in the Table 4.1. The regression

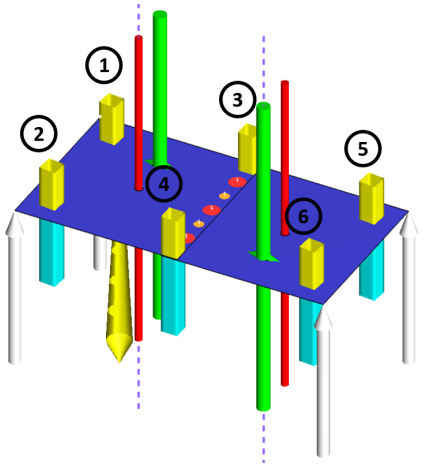


Figure 4.2: Joining station.

SVR penalty, C	SVR epsilon tube, ϵ	Ridge, α	RBF bandwidth, σ
1E2	1E-2	1E-3	1E-1

Table 4.1: Regression parameters.

models were fitted using 17 data test points with clamping position as a variation and the mean deviation of geometry as a response of geometry. The metamodels are presented on Fig. 4.3 and 4.4 as a blue line. Here the clamp position is plotted against the mean deviation. The dark blue dots is the data which was used for fitting. Red points are predicted by metamodel test points, while green ones are the simulated results for the same design variables. From the Table 4.2 the assessment of the metamodels can be done. Both of them return good approximations. The SVR model returns smaller errors compared to Ridge regression, but Ridge model has better coefficient of determination. The second part can be explained by the epsilon parameter of the SVR which gives some freedom to the model. That can be seen from the coefficient of determination R^2: it is closer to 1 in case of the Ridge regression, as the model tries to go through all the points. Additionally, there is one testing point which is on the boundary of the metamodel, the Ridge metamodel deals with it better in this case. Generally, points outside the metamodel's fitting region should not be used for prediction.

Regression	MSE	AAE	MAE	R^2
SVR	0.00087	0.02484	0.05282	0.96041
Ridge	0.00106	0.02560	0.05472	0.96594

Table 4.2: Regression evaluation.

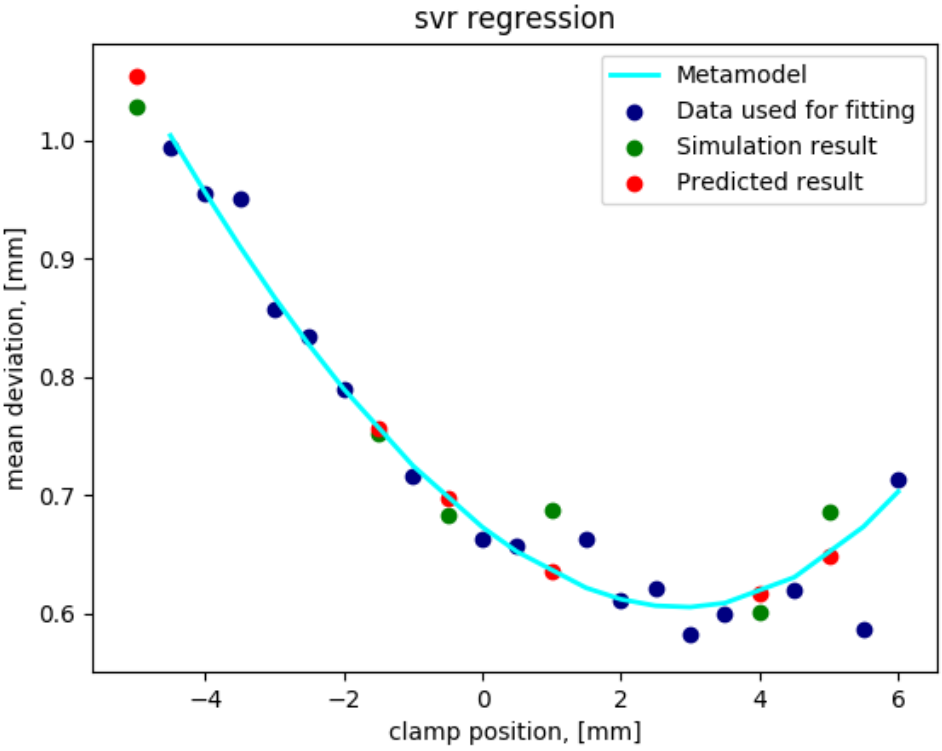


Figure 4.3: SVR model built using one parameter.

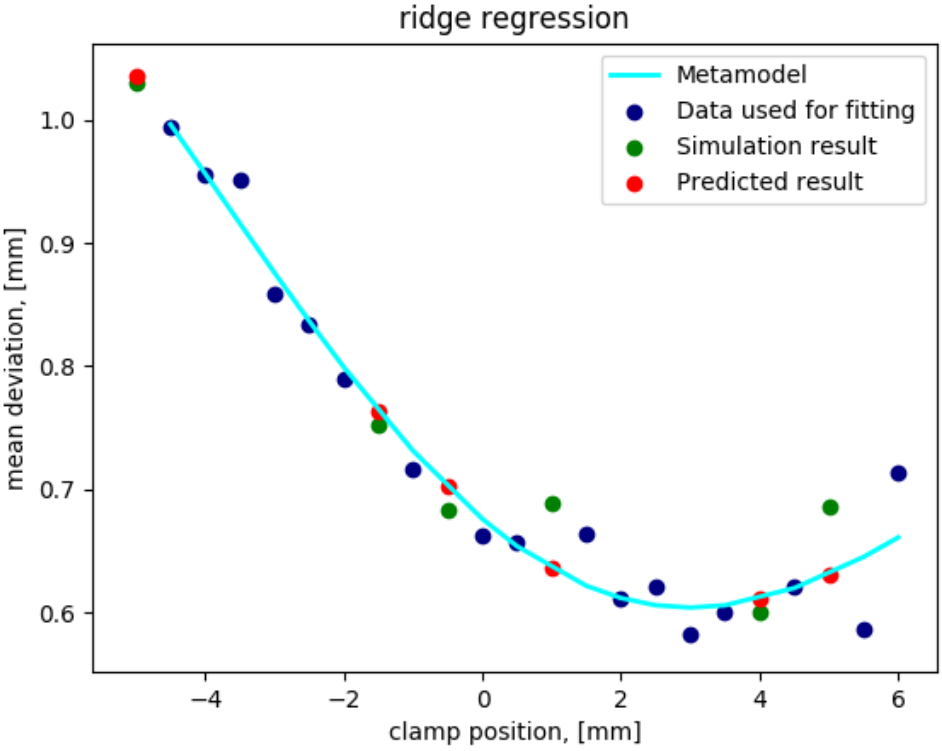


Figure 4.4: Ridge regression built using one parameter.

4.3 Two design variables metamodel

For the second investigation two clamps were varied: now clamps number 2 and 3 (Fig. 4.2) were varied only in z direction (shimsing). The clamps were chosen again arbitrary. The same range was used: from -5 mm to 5 mm from initial location with step of 0.5 mm. 195 out of 200 cases converged. After the exclusion of the repeated data sets, 127 cases were used for fitting and 30 for the evaluation of the regression.

Different regression parameters were checked again, starting from the one performed best in the first test. The comparison can be seen in App. D. The best parameters are close to the ones used in the previous test: only the penalty C for the SVR is smaller.

The metamodels are presented on the Fig. 4.5 and 4.6 on the two upper most figures. Two clamp locations are plotted there against the mean deviation of the geometries, which is additionally shown with the color. The points used for the building of the regression can be seen in blue, as well as the testing points (green simulated and red predicted). There are also numbered cases, which present the resulted from the simulation geometries to illustrate effects of the clamp relocation on the outcome assembly. The plots present worst case (74), best case (46) and average (146). The initial imperfect input geometry is also demonstrated on the figure. Each point in the pointcloud on the metamodel plots represents the specific outcome of the joining simulation. The surface is therefore represents all of these outcomes as a mathematical interpolation function of two clamping locations.

The evaluation of the regression is introduced in the Table 4.3. The quality of the metamodels are similar, but it is worse comparing to the one variable metamodels. The possible improvement of the metamodel by changing the sampling technique is therefore investigated in the next Section.

Regression	MSE	AAE	MAE	R^2
SVR	0.00472	0.05680	0.13359	0.71727
Ridge	0.00479	0.05875	0.11277	0.73609

Table 4.3: Regression evaluation.

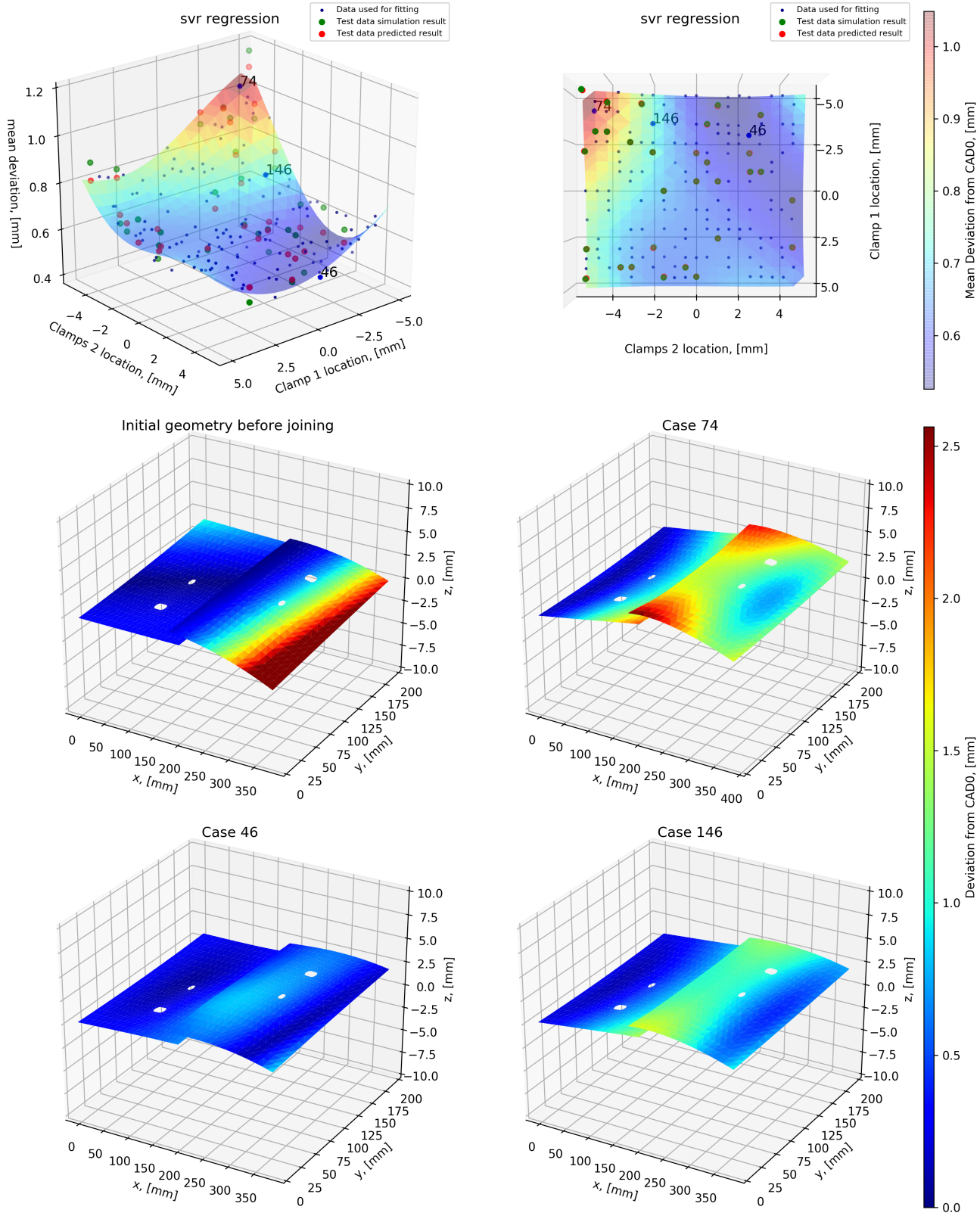


Figure 4.5: SVR model built using two parameters.

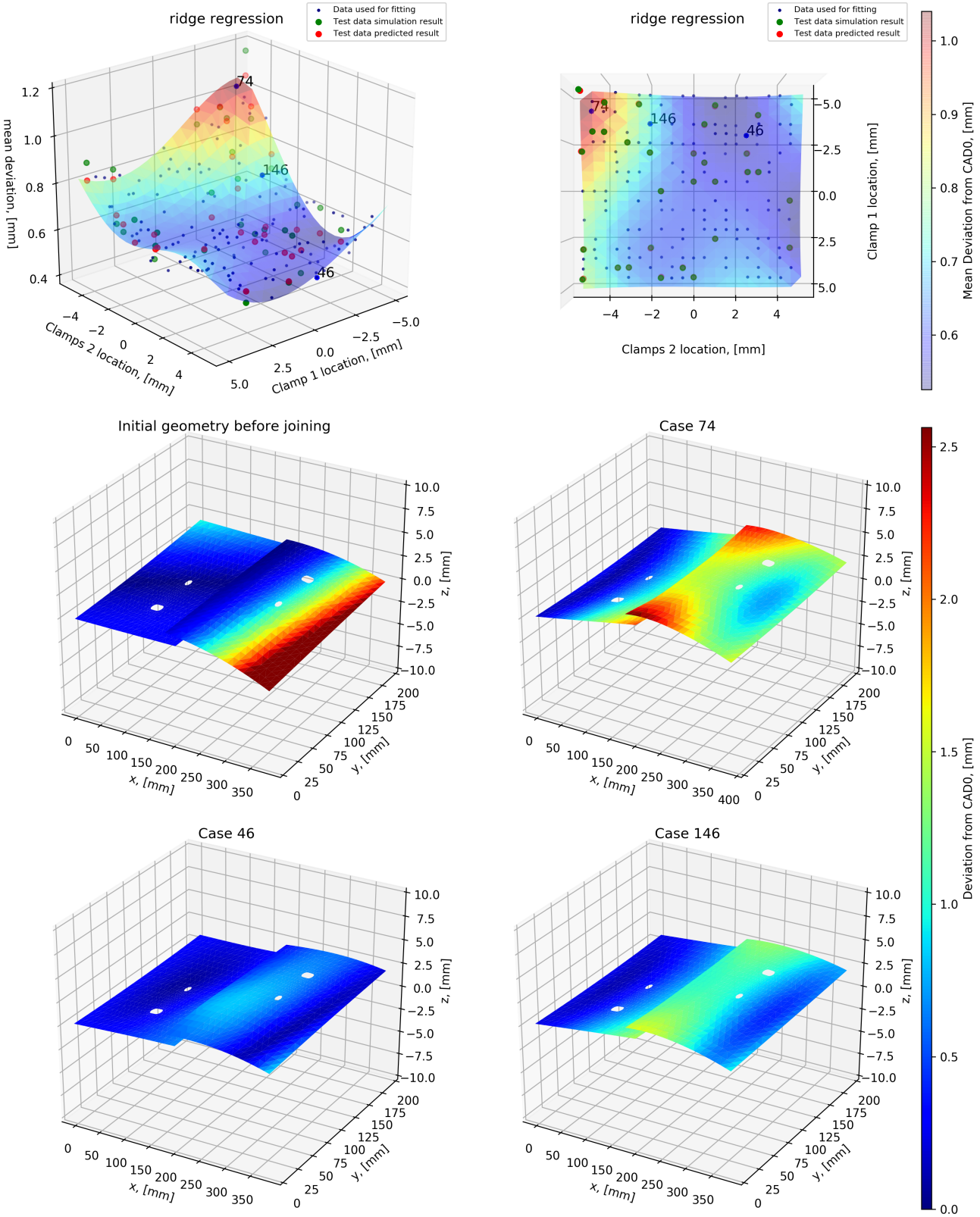


Figure 4.6: SVR model built using two parameters.

4.4 Two design variables metamodel using LHC

To improve the metamodel while reducing computational cost the controlled sampling technique should be considered. In the random uniform sampling technique the data points are randomly picked, which might result in repeated or concentrated in specific area data sets. The LHC technique creates points, which cover the whole parameter space - see Sec. 2.4.4.

Same set up was checked to produce the metamodel with similar physics behind it. About same amount of samples were produced and solved using LHC - 156: 126 were used for the fitting and 30 for evaluation. Additionally bigger sample set was taken to check the improvement of the model (220 samples with 190 for fitting and 30 for checking). Only SVR regression was constructed, as Ridge and SVR returned similar results in previous investigations.

The metamodels can be seen on Fig. 4.7, all of them return similar looking predictions. If to check the errors (Table 4.4), both LHC based on similar and larger data sizes return worse results. It might be explained by the better spread testing set which 'caught' more outliers. Additionally, in uniform random sampling technique the cases were created using the steps, the values in between were not checked. As it can be seen on the first row of the Fig. 4.7, there are several contributions to the metamodel for the identical locations of the same clamp. In case of LHC each design point is present only once, which is supposed to give more information about the response returning better approximation. More testing should be done to investigate the better implementation of LHC or other sampling techniques.

Sampling	MSE	AAE	MAE	R^2
Random 150	0.00472	0.05680	0.13359	0.71727
LHC 150	0.01024	0.07640	0.28245	0.60730
LHC 220	0.00959	0.07220	0.28732	0.77248

Table 4.4: SVR Regression evaluation using different sampling.

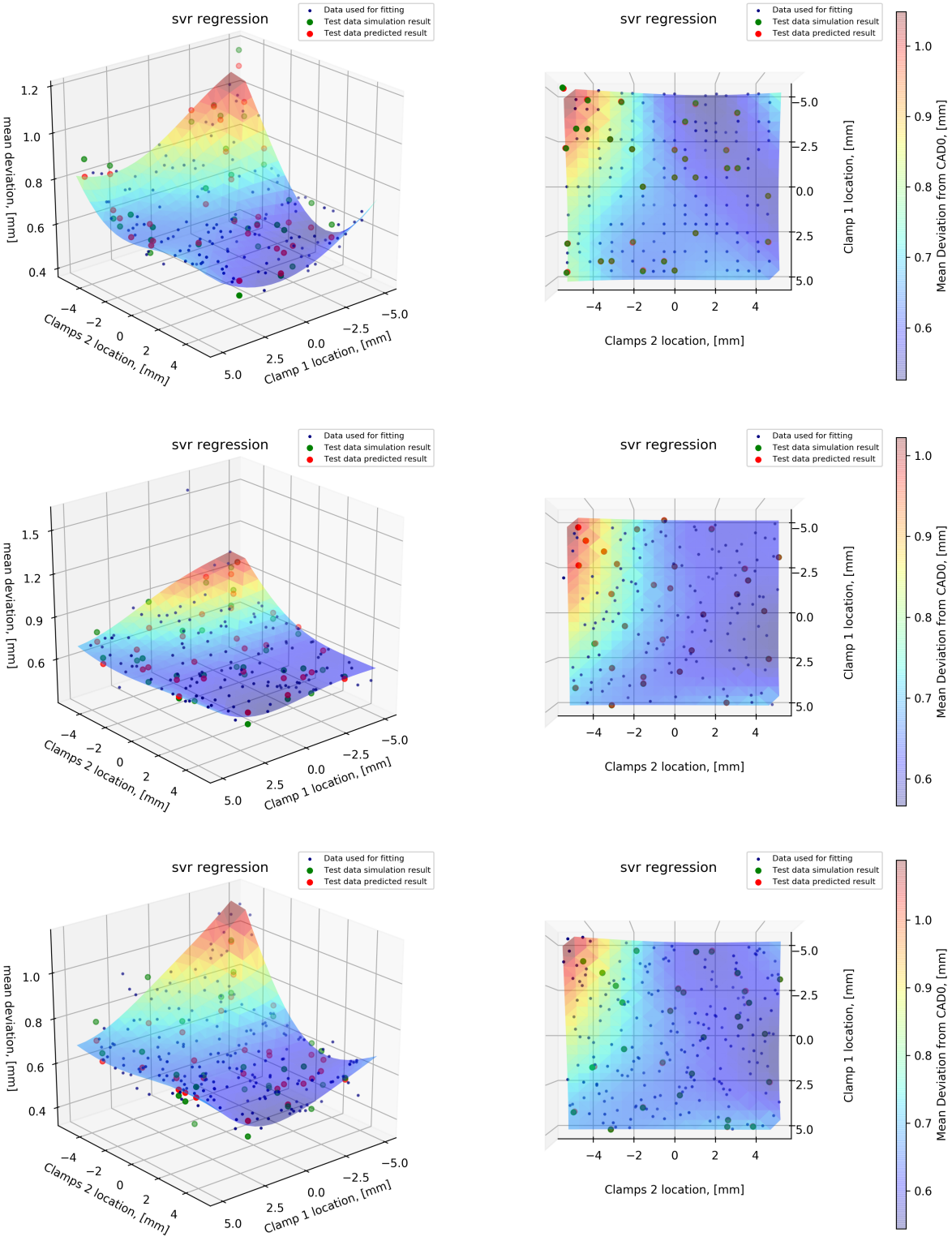


Figure 4.7: SVR Regressions using top to bottom: Random sampling with 150 cases, LHC sampling with 150 cases and LHC with 220 cases.

4.5 18 design variables metamodel

Additionally the investigation with varying six clamps in x, y and z was performed (Fig. 4.2). It resulted in 18 design variables. Range of variation for x and y was from -10 to 10 mm. The actual z locations were not varied, as the part is near flat and the DFS tool during clamp generation moves it back (See Sec. 3.3.1 for more explanation). As in previous tests shimsing was set to vary to produce the variation in z in the range -5 to 5 mm.

Here the problem in setting the experiment is that a part of the simulations crash. The percentage of the non-converged simulations is growing with the complexity of the model - both the simulation model and regression model. Here 1950 random cases were created, out of them 1300 converged (66%). To compare with the test case with two design variables - about 97% of the simulations converged there. What is more, the computational time for each case became longer - more steps had to be computed to reach the equilibrium, as most of the random clamping resulted in unbalanced set up.

In the Table 4.5 the assessment of the metamodels is presented. In this case the SVR return better approximation, although the quality is reduced comparing with previous metamodels. Bigger sample size has to be tested in the future.

Regression	MSE	AAE	MAE	R^2
SVR	0.29707	0.45151	1.14719	0.80744
Ridge	0.42063	0.53298	1.67276	0.83283

Table 4.5: SVR and Ridge regressions errors for the 18 design variables regression models.

Chapter 5

Discussion

Comparing to the most of the state-of-the-art examples (See Sec. 1.4) the presented study did not include the optimization goal, however also pursuing the accurate approximation metamodel of the nonlinear problem. What is more, as most of the studies the model was built using data received from the numerical simulations.

The metamodels with one and two design variables return very good approximation. They do not require many samples and can be plotted. The figures provide better understanding of the effects in the model. The line/surface represents the response of the geometry to the change of the clamping. The metamodel is therefore includes physical effects replaced by mathematical function. It gives an opportunity to interpolate between the results and to choose the suitable design of the joining station. As the elastic model is used for the parts, there is no plasticity involved in the process. Therefore, it can be noted that the clamps create only the moment to balance out the imperfections during the joining, but do not deform the part itself.

As the complexity of the model is increasing, both geometry and variable vise, the computation times for solving is growing. Another problem here is that with more complex geometry of the parts more of the simulations do not converge. The most common reason is that the equilibrium cannot be reached. The solution for that is smaller variation range of clamps or keeping some of them fixed.

Most of the literature studies ([34, 20, 9]) have chosen SVR parameters according to Cherkassky and Ma [5], while in the presented study different parameters were compared through tests by error assessment.

In reduction of computation time by different sampling strategies Latin Hypercube did

not prove to be more efficient nor to produce better approximation models in the current study. This outcome is similar to observation in the study by Andolfatto et al. [3], where Latin Hypercube did not improve the approximation quality. Another issue here is that the locations defined by the variation in the automated environment are not the true clamp locations (Sec. 3.3.1), which makes it hard to apply the controlled sampling technique (LHC). In the presented prototype the part is near flat, which does not cause problem. Variation in z direction is simply replaced by shimsing. In the case of the real car parts, the geometries contain curvatures, which makes it hard to move along the surface.

The investigation has shown that the Support Vector Regression and Kernel Ridge regression produce similar metamodels. SVR produces better approximation model with more design variables comparing to Kernel Ridge regression. It can be explained by epsilon tube parameter of the regression. When the data points are inside the tube, they do not have contribution to the penalty function. Therefore there is an error which is acceptable, which result in more freedom of the model. That results in smaller average errors along the whole model.

The initial station set up should be predefined. In the future the numerical optimization can assist in defining the optimum for the CAD0 set up, which can be then used as a starting point for the variation. Another important issue is that the range of variation for each clamp and for surface has to be defined by user.

Chapter 6

Summary

To define a perfect set up of the joining station for each imperfect geometry case will result in large number of numerical simulation. Each imperfect part in the production will result in one case of numerical optimization. Other methods were studied to investigate the possibility of creation of metamodel, where all the physical effects are taken into account.

During the project the automated simulation environment was programmed. It assisted in controlled creation, solution and analysis of the joining cases. The general idea was to study the possibility of interpolation between varying joining station - both in geometry of the joined parts and station set up.

The theory behind the simulation, FEM and computational contact mechanics was studied then to present the behavior of the model, which is highly nonlinear. The different techniques for metamodeling were compared and the suitable ones were picked for investigation. Support Vector Regression and Kernel Ridge Regression were chosen based on the previous experiments presented in literature.

First the simple model with one design variable was tested, different regression parameters were compared and the best fit was found basing on the error estimation. As the test was successful the problem was expanded: two design variables were used. More computational effort was needed for solving of the cases. The investigation was successful, although the error has grown comparing to the first test. To improve the metamodel while reducing the computation times a different sampling technique was studied (Latin Hypercube). It did not prove to improve the results, therefore further studies should be performed here. The metamodeling techniques were tested also on the larger problem size, 18 design variables were used. Two challenges appeared for this problem size. Large

part of the created cases did not converge as well as the computational time for each case became longer.

In general, the possibility of metamodel implementation to reduce the number of simulations was proved to be successful. More data for the more complex problems is needed, as well as better sampling technique should be utilized. Different metamodeling techniques should be tested as well, such as Artificial Neural Networks.

Chapter 7

Future work

As the Automated Simulation Environment was able to assist with simulation of simplified sheet metal assemblies, the case creation module was checked to function as well with more complicated models, such as automobile door (See Fig. 7.1). The future of the Automation Simulation environment is to produce simulation cases for different joining techniques, such as clinching, roller hemming, welding for Body in White of automobile. The future of the project is not only to increase the number of parameters in the joining station, but also interpolate between different variations of the imperfections in the parts. It is planned to create the metamodel with the help of artificial intelligence, which will be trained to assist with selection of the best joining station for certain imperfect geometry. The results of the selection should be least deviated from CAD0 assembly. The ideal future scenario will be to scan or measure the 'real' parts before the joining process, use them as an input for the metamodel and receive the perfect joining set up for the part.

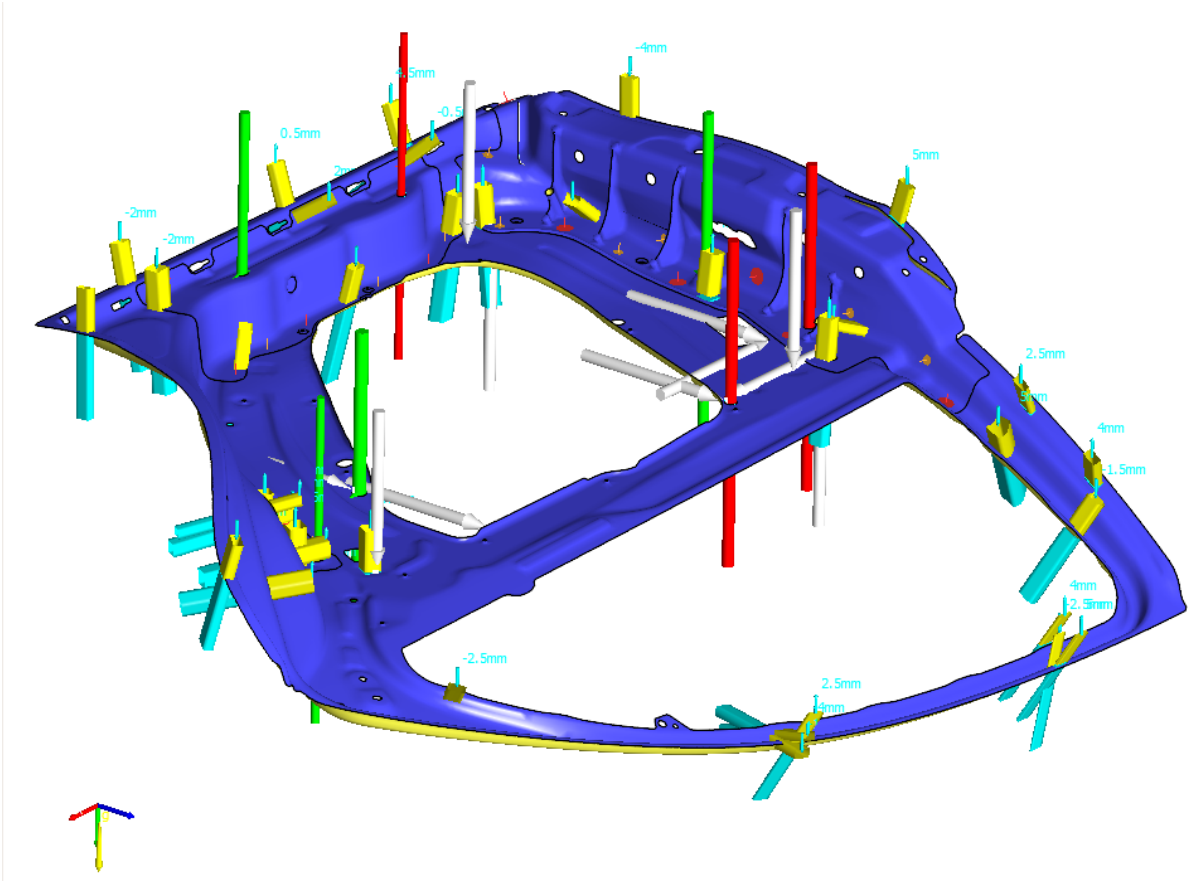


Figure 7.1: Varied clamping as a result of the Automated Environment.

Bibliography

[1] *LS-DYNA Theory Manual*.

[2] Scikit-learn: Machine learning in Python. URL http://scikit-learn.org.

[3] Loïc Andolfatto, François Thiébaut, Marc Douilly, and Claire Lartigue. On neural networks' ability to approximate geometrical variation propagation in assembly. *Procedia CIRP*, 10:224 – 232, 2013. ISSN 2212-8271. doi: http://dx.doi.org/10.1016/j.procir.2013.08.035. URL http://www.sciencedirect.com/science/article/pii/S2212827113005842. The Twelfth CIRP Conference on Computer Aided Tolerancing.

[4] Russell R. Barton and Martin Meckesheimer. Chapter 18 metamodel-based simulation optimization. *Handbooks in Operations Research and Management Science*, 13:535 – 574, 2006. ISSN 0927-0507. doi: http://dx.doi.org/10.1016/S0927-0507(06)13018-2. URL http://www.sciencedirect.com/science/article/pii/S0927050706130182. Simulation.

[5] Vladimir Cherkassky and Yunqian Ma. Practical selection of svm parameters and noise estimation for svm regression. *Neural Networks*, 17(1):113 – 126, 2004. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(03)00169-2. URL http://www.sciencedirect.com/science/article/pii/S0893608003001692.

[6] Livemore Software Technology Corporation. Ls-dyna, 2011. URL http://www.lstc.com/products/ls-dyna.

[7] Stefan Dahlström and Lars Lindkvis. Variation simulation of sheet metal assemblies using the method of influence coefficients with contact modeling. *Journal of Manufacturing Science and Engineering*, 129(3):615 – 622, 2006. doi: http://dx.doi.org/10.1115/1.2714570.

[8] Abhishek Das, Pasquale Franciosa, David Williams, and Darek Ceglarek. Physics-driven shape variation modelling at early design stage. *Procedia CIRP*, 41: 1072 – 1077, 2016. ISSN 2212-8271. doi: http://dx.doi.org/10.1016/j.procir.2016.01.031. URL http://www.sciencedirect.com/science/article/pii/S2212827116000457. Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems.

[9] S. Dey, T. Mukhopadhyay, and S. Adhikari. Metamodel based high-fidelity stochastic analysis of composite laminates: A concise review with critical comparative assessment. *Composite Structures*, 171:227 – 250, 2017. ISSN 0263-8223. doi: http://dx.doi.org/10.1016/j.compstruct.2017.01.061. URL http://www.sciencedirect.com/science/article/pii/S0263822316328793.

[10] Hugo Falgarone, François Thiébaut, Jérôme Coloos, and Luc Mathieu. Variation simulation during assembly of non-rigid components. realistic assembly simulation with anatoleflex software. *Procedia CIRP*, 43:202 – 207, 2016. ISSN 2212-8271. doi: http://dx.doi.org/10.1016/j.procir.2016.02.336. URL http://www.sciencedirect.com/science/article/pii/S2212827116006314. 14th CIRP CAT 2016 - CIRP Conference on Computer Aided Tolerancing.

[11] Python Software Foundation. Python. URL https://www.python.org/about/.

[12] Salvatore Gerbino, Pasquale Franciosa, and Stanislao Patalano. Parametric variational analysis of compliant sheet metal assemblies with shell elements. *Procedia CIRP*, 33:339 – 344, 2015. ISSN 2212-8271. doi: http://dx.doi.org/10.1016/j.procir.2015.06.077. URL http://www.sciencedirect.com/science/article/pii/S2212827115007209. 9th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '14.

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[14] Lazhar Homri, Edoh Goka, Guillaume Levasseur, and Jean-Yves Dantan. Tolerance analysis — form defects modeling and simulation by modal decomposition and

optimization. *Computer-Aided Design*, 91:46 – 59, 2017. ISSN 0010-4485. doi: http://dx.doi.org/10.1016/j.cad.2017.04.007. URL http://www.sciencedirect.com/science/article/pii/S0010448517300623.

[15] Henrique M. Kroetz, Rodolfo K. Tessari, and André T. Beck. Performance of global metamodeling techniques in solution of structural reliability problems. *Advances in Engineering Software*, 2017. ISSN 0965-9978. doi: https://doi.org/10.1016/j.advengsoft.2017.08.001. URL http://www.sciencedirect.com/science/article/pii/S0965997817305331.

[16] W. Michael Lai, David Rubin, and Erhard Krempl. *Introduction to Continuum Mechanics (4th Edition)*. Elsevier, 2010. ISBN 978-0-7506-8560-3. URL http://app.knovel.com/hotlink/toc/id:kpICME0015/introduction-continuum/introduction-continuum.

[17] Y.F. Li, S.H. Ng, M. Xie, and T.N. Goh. A systematic comparison of metamodeling techniques for simulation optimization in decision support systems. *Applied Soft Computing*, 10(4):1257 – 1273, 2010. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2009.11.034. URL http://www.sciencedirect.com/science/article/pii/S1568494609002841. Optimisation Methods and Applications in Decision-Making Processes.

[18] Bob McGinty. Continuum mechanics. URL http://www.continuummechanics.org/.

[19] Kevin P. Murphy. *Machine Learning : A Probabilistic Perspectives*. MIT Press, 2014. ISBN 9780262305242.

[20] Mahdi Arian Nik, Kazem Fayazbakhsh, Damiano Pasini, and Larry Lessard. A comparative study of metamodeling methods for the design optimization of variable stiffness composites. *Composite Structures*, 107:494 – 501, 2014. ISSN 0263-8223. doi: http://dx.doi.org/10.1016/j.compstruct.2013.08.023. URL http://www.sciencedirect.com/science/article/pii/S0263822313004200.

[21] A. Olsson, G. Sandberg, and O. Dahlblom. On latin hypercube sampling for structural reliability analysis. *Structural Safety*, 25(1):47 – 68, 2003. ISSN 0167-

4730. doi: https://doi.org/10.1016/S0167-4730(02)00039-5. URL http://www.sciencedirect.com/science/article/pii/S0167473002000395.

[22] Binbin Pan, Guangming Zhang, James J. Xia, Peng Yuan, Horace H.S. Ip, Qizhen He, Philip K.M. Lee, Ben Chow, and Xiaobo Zhou. Prediction of soft tissue deformations after cmf surgery with incremental kernel ridge regression. *Computers in Biology and Medicine*, 75(Supplement C):1 – 9, 2016. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2016.04.020. URL http://www.sciencedirect.com/science/article/pii/S0010482516301093.

[23] Hong-Seok Park and Xuan-Phuong Dang. Structural optimization based on cad–cae integration and metamodeling techniques. *Computer-Aided Design*, 42(10):889 – 902, 2010. ISSN 0010-4485. doi: http://dx.doi.org/10.1016/j.cad.2010.06.003. URL http://www.sciencedirect.com/science/article/pii/S0010448510001107.

[24] Giovanni Pizzi, Andrea Cepellotti, Riccardo Sabatini, Nicola Marzari, and Boris Kozinsky. Aiida: automated interactive infrastructure and database for computational science. *Computational Materials Science*, 111:218 – 230, 2016. ISSN 0927-0256. doi: http://dx.doi.org/10.1016/j.commatsci.2015.09.013. URL http://www.sciencedirect.com/science/article/pii/S0927025615005820.

[25] Mike Puso, Tod Laursen, and Jerome Solberg. A segment-to-segment mortar contact method for quadratic elements and large deformations. 197:555–566, 01 2008.

[26] Roland Rosen, Georg von Wichert, George Lo, and Kurt D. Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 48(3):567 – 572, 2015. ISSN 2405-8963. doi: http://dx.doi.org/10.1016/j.ifacol.2015.06.141. URL http://www.sciencedirect.com/science/article/pii/S2405896315003808. 15th IFAC Symposium on Information Control Problems in Manufacturing.

[27] Benjamin Schleich and Sandro Wartzack. Challenges of geometrical variations modelling in virtual product realization. *Procedia CIRP*, 60:116 – 121, 2017. ISSN 2212-8271. doi: http://dx.doi.org/10.1016/j.procir.2017.01.019. URL http://www.sciencedirect.com/science/article/pii/S2212827117300203. Complex

Systems Engineering and Development Proceedings of the 27th CIRP Design Conference Cranfield University, UK 10th – 12th May 2017.

[28] Benjamin Schleich, Nabil Anwer, Luc Mathieu, and Sandro Wartzack. Shaping the digital twin for design and production engineering. *CIRP Annals - Manufacturing Technology*, 66(1):141 – 144, 2017. ISSN 0007-8506. doi: http://dx.doi.org/10.1016/j.cirp.2017.04.040. URL http://www.sciencedirect.com/science/article/pii/S0007850617300409.

[29] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, Aug 2004. ISSN 1573-1375. doi: 10.1023/B:STCO.0000035301.49549.88. URL https://doi.org/10.1023/B:STCO.0000035301.49549.88.

[30] Georg Ungemach and Frank Mantwil. Efficient consideration of contact in compliant assembly variation analysiss. *Journal of Manufacturing Science and Engineering*, 131 (1):011005–011005–9, 2008. doi: http://dx.doi.org/10.1115/1.3046133.

[31] Hu Wang, Enying Li, and Guang Yao Li. The least square support vector regression coupled with parallel sampling scheme metamodeling technique and application in sheet forming optimization. *Materials and Design*, 30(5):1468 – 1479, 2009. ISSN 0261-3069. doi: https://doi.org/10.1016/j.matdes.2008.08.014. URL http://www.sciencedirect.com/science/article/pii/S0261306908004135.

[32] P. Wriggers. *Computational Contact Mechanics*. Springer Berlin Heidelberg, 2006. ISBN 9783540326090.

[33] P. Wriggers. *Nonlinear Finite Element Methods*. Springer-Verlag Berlin Heidelberg, 2008. ISBN 9783540710004.

[34] Ping Zhu, Feng Pan, Wei Chen, and Siliang Zhang. Use of support vector regression in structural optimization: Application to vehicle crashworthiness design. *Mathematics and Computers in Simulation*, 86:21 – 31, 2012. ISSN 0378-4754. doi: http://dx.doi.org/10.1016/j.matcom.2011.11.008. URL http://www.sciencedirect.com/science/article/pii/S0378475412000936. The Seventh International Symposium on Neural Networks + The Conference on Modelling and Optimization of Structures, Processes and Systems.

Appendices

Appendix A

Python libraries

- `numpy`,`scipy` for powerful scientific computing
- `argparse` for creating of user friendly command line interface
- `os`, `subprocess`, `shutil` for accessing the OS system and files
- `stl` for creation of STL mesh files
- `lxml` for pasing of xml files
- `random`, `pyDOE` to create the experiment both by Random and Latin Hypercube sampling
- `sklearn` for building of SVR and Kernel Ridge Regression
- `matplotlib` for plotting
- `re`, `math`, `itertools`, `copy`

Appendix B

Input for the automated environment

Each of the parts of the framework (`create`, `solve`, and `analyze`) is called through the command line in shell. Some of the arguments are specified in the command line, some of them inside of the separate module.

To run `create` we write:

```
python create.py pfile number_geom_cases number_station_cases
number_test_cases --path_xml --path_cases
```

here `pfile` is the name of project xml file, `number_geom_cases` is the number of imperfect geometries' sets created, `number_station_cases` is number of clamping cases for each geometry case, `number_test_cases` number of additional cases used for solving, `--path_xml` path to xml file, `--path_cases` path, where the cases folder will be stored.

To run `solve` we submit the command:

```
python solve.py path --folder
```

with `path` as a location of the folder and `--folder` name of the folder.

To run `analyze` we write:

```
python analyze.py path test_size --plot
```

with `path` as a location of the folders with solved cases, `test_size` number of cases to use for the testing of the regression and `--plot` as a command to plot each of the resulting geometry with the deviation from `CAD0` as a color.

A separate module was also created to easily define and change all necessary variables of the program, as well as to easily use it along the whole program. It contains 5 dictionaries:

- `VARIATION_RANGES` with all the ranges to define the geometry and clamp variation
- `ACPROX_KEYWORDS_GER` with all the keywords used in XML tree file
- `ACPROX_COMMANDS` containing all the command used for DFS batch
- `DYNA_KEYWORDS` with LS-DYNA keywords
- `PATHS` with all the files and tool locations

Each of the dictionaries provide an easy access to variables by key words. In this way, modifications of variables can be easily performed in one place.

Appendix C

Files for Data Base

- `project_file.ACPROX` to have a possibility to resolve the case if need arises
- `station_parameters.txt` for the future analysis of the cases, parameters are extracted from `project_file.ACPROX` and `input.key` (the reason for the use of the second one is that real locations of the clamps can be extracted only from the mesh - see Sec. 3.3.1), each of the parameters is noted by key in form `*key_name` (similar to LS-DYNA style)
- `Initial_geo.stl` STL format geometry file with geometry used for simulation, extracted from `input.key` file. See Sec. 3.3.3 for explanation how the mesh is transformed in STL
- `Resulting_geo.stl` STL format geometry file with geometry resulting from the simulation, extracted from `Result.geo` file.
- `Result.geo` to have the resulting geometry in LS-DYNA format, to have a possibility to use this geometry for next step simulations if needed
- `Result.pst` to have the resulting stresses if additional stress analysis is needed

Appendix D

Comparison of regression parameters

Here the comparison for the different parameters for the Regression are presented. The measured by different criteria, such multiple error types of the testing data set and coefficient of determination. The comparison was done for the model with one design variable. The best parameters were chosen when the most of the criterias were satisfied - yellow columns in Tables D.1, D.2, D.3 and D.4. For definition of errors see Eq. (4.1.1)-(4.1.4). The comparison was done for both tests with one (D.1, D.2) and two (D.3, D.4) design variables. In the current study there is no significant influence of parameter change while adding the dimension. That could be explained same influence of the variable on the model - both of the clamps are moved in z direction.

SVR						
Parameters	ϵ	1.00E-01	1.00E-02	1.00E-02	1.00E-02	1.00E-03
	C	1.00E+00	1.00E+00	1.00E+01	1.00E+02	1.00E+02
Errors	MSE	0.00588	0.00105	0.00088	0.00087	0.00092
	AAE	0.07019	0.02708	0.02304	0.02484	0.02374
	MAE	0.12442	0.05670	0.05406	0.05282	0.05154
	R^2	0.71260	0.96343	0.96234	0.96041	0.96498

Table D.1: Comparison of parameters used for SVR with one design variable.

Kernel Ridge Regression					
Parameters	α	1.00E-01	1.00E-02	1.00E-03	1.00E-04
Errors	MSE	0.00373	0.00118	0.00106	0.00117
	AAE	0.04888	0.02941	0.02560	0.02720
	MAE	0.09940	0.05771	0.05472	0.06144
	R2	0.87716	0.96196	0.96594	0.97006

Table D.2: Comparison of parameters used for Kernel Ridge Regression with one design variable.

SVR				
Parameters	ϵ	1.00E-02	1.00E-02	1.00E-03
	C	1.00E+02	1.00E+01	1.00E+01
Errors	MSE	0.00506	0.00472	0.00474
	AAE	0.05752	0.05680	0.05640
	MAE	0.14186	0.13359	s 0.13681
	R^2	0.71640	0.71727	0.71739

Table D.3: Comparison of parameters used for SVR with two design variables.

Kernel Ridge Regression				
Parameters	α	1.00E-02	1.00E-03	1.00E-04
Errors	MSE	0.00519	0.00479	0.00481
	AAE	0.06227	0.05875	0.05813
	MAE	0.12446	0.11277	0.13877
	R2	0.72160	0.73609	0.75846

Table D.4: Comparison of parameters used for Kernel Ridge Regression with two design variables.